Hello! world

JSP/Servlet

임명진 연구원

스마트인재개발원
Smart Human Resources Development

Contents

1. 서버프로그램의 필요성을 설명할 수 있다.
2. WAS 구조를 설명할 수 있다.
3. Servlet의 특징을 설명할 수 있다.

네트워크(network)

네트워크(network)란 무엇일까?

네트워크(network)

network = net + work

"그물처럼 서로 엮여서 일하는 것"

통신 장비들이 그물망처럼 연결되어 데이터를 교환하는 형태(통신망)

네트워크(network)

Node

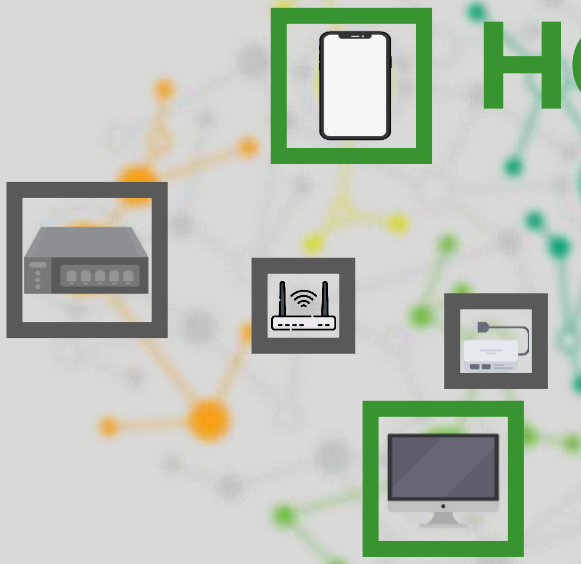
네트워크에 연결된 컴퓨터와
그 안에 속한 장비
(허브(HUB), 공유기, 라우터 …)

HUB : 다수의 PC와 장치들을 묶어서 LAN 구성 시 사용하는 장치

라우터 : 데이터를 주고 받을 때 가장 적절한 통신통로를 이용해 전송해주는 장치

공유기 : 하나의 인터넷 라인을 공유해 동시에 인터넷에 접속할 수 있도록 하는 장치

네트워크(network)



HOST 노드 중에서 애플리케이션을 실행할 수 있는 컴퓨팅 시스템을 갖춘 기기

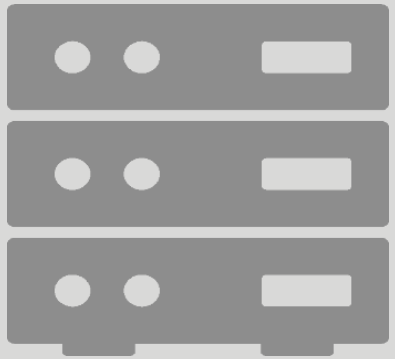
Client와 Server

Host 사이에 제공되는 서비스를 기준으로 Host를 세분화

Client

→ 서비스를 요청하고 사용하는 host

서비스를 제공(응답)하는 host ←

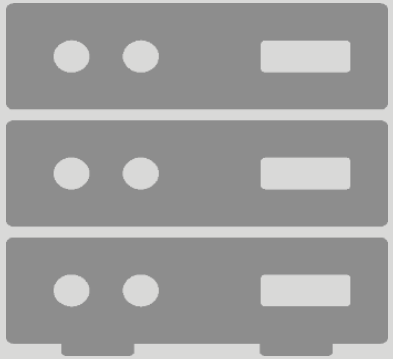


Server

Client와 Server

서비스를 요청하고 사용하는 host

서비스를 제공하는 host



Client

Server

각 호스트는 클라이언트나 서버로 고정되지 않고 이용하는 서비스의 종류에 따라 클라이언트가 될 수도 있고 서버가 될 수도 있음

Client와 Server

네이버에 스마트인재개발원을
검색하고 싶어

?

기상청

Client와 Server

네이버에 스마트인재개발원을
검색하고 싶어

→ Server

Client와 Server

?

오늘 광주 날씨

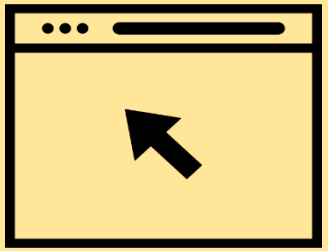
기상청

Client와 Server

Client

오늘 광주 날씨

Client와 Server



Web Browser

Client

요청(request)

HTTP

응답(response)

정적리소스 (HTML, XML …)

Web Server

Server

Web Browser

웹 서버와 통신하여 html, css, js, text, 이미지 등 웹 리소스를 받아
사용자에게 보여주는 그래픽 사용자 인터페이스(GUI) 기반의 소프트웨어

Web Server

HW :

웹 서버의 소프트웨어와 웹 사이트의 컴포넌트 파일들을 저장하는 장비

SW :

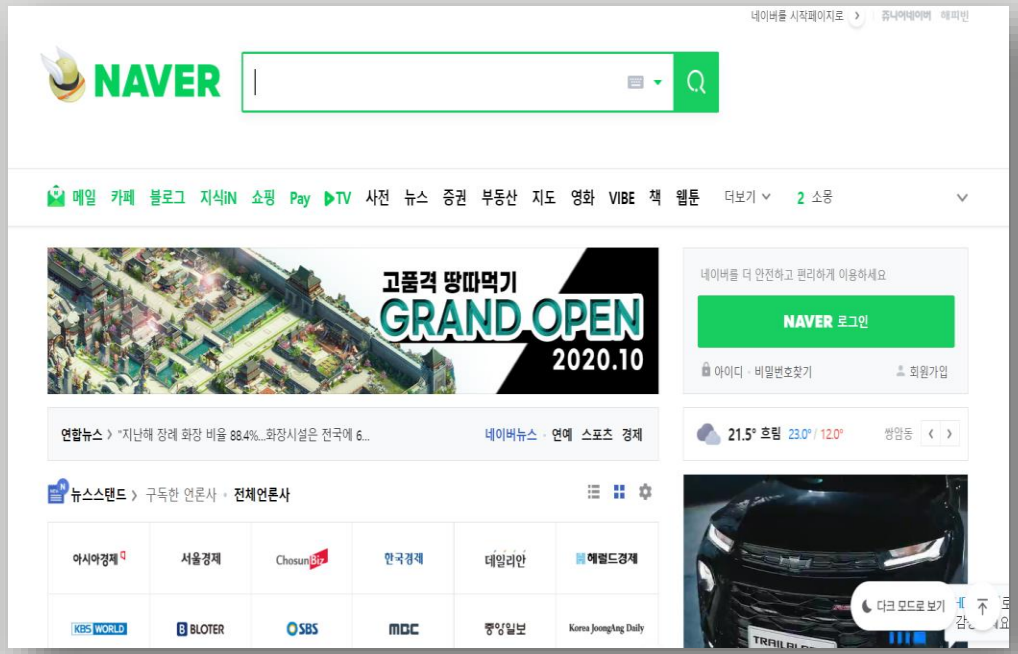
클라이언트로부터 HTTP 프로토콜로 요청을 받아 응답해주는 프로그램

Web Server

정적웹서버 :
HTTP 서버(SW) 가 있는 컴퓨터(HW)로 구성

서버에 저장되어 있는 파일 (매번 가공X)을
클라이언트에 전송

서버 프로그램의 필요성



두 페이지의 차이점이 무엇일까요?

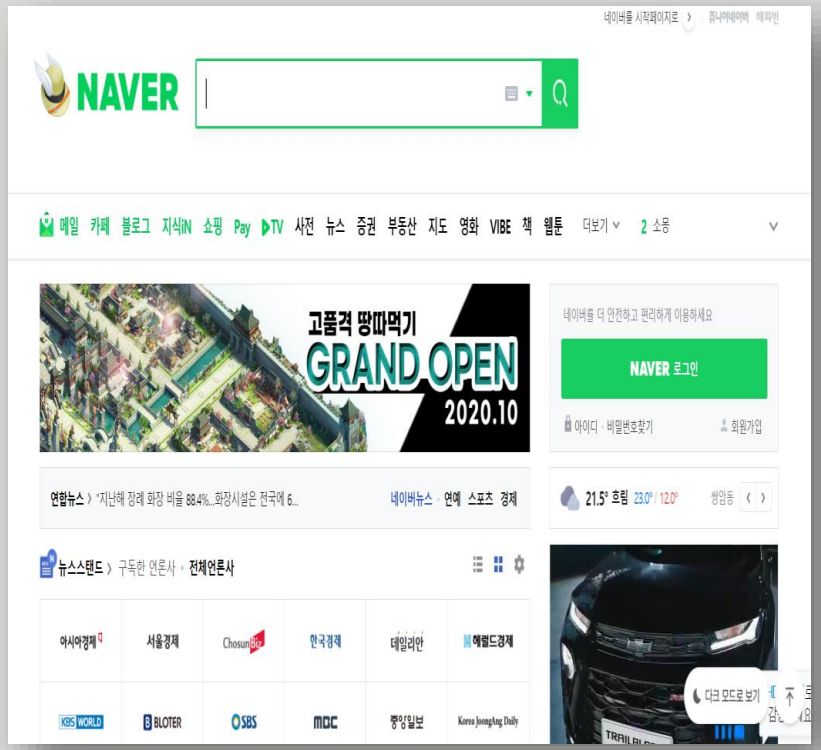
서버 프로그램의 필요성

정적페이지 (Static Web Page) :

서버(웹 서버)에 미리 저장된 파일(HTML 파일, 이미지, Javascript 파일 등)이 그대로 전달되는 웹페이지

사용자는 서버에 저장된 데이터가 변경되지 않는 한 고정된 웹 페이지를 보게 됨

서버 프로그램의 필요성



동적페이지 (Dynamic Web Page) :

서버(웹 서버)에 있는 데이터들을 스크립트에 의해
가공처리한 후 생성되어 전달되는 웹페이지

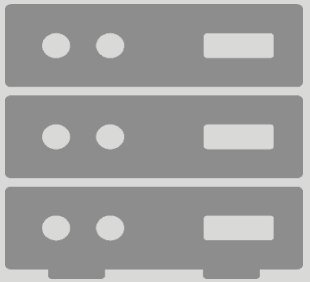
사용자는 상황, 시간, 요청 등에 따라 달라지는
웹 페이지를 보게 됨

웹 통신

Client

위키백과 메인 페이지 보고 싶어!

https://ko.wikipedia.org/



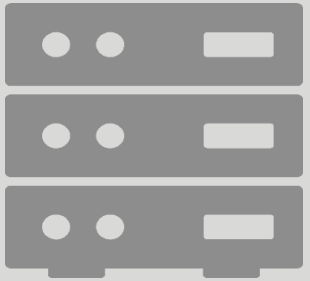
Server

웹 통신

위키백과 메인 페이지 보고 싶어!

https://ko.wikipedia.org/

Client



Server

저장하고 있던
정적리소스 (HTML)

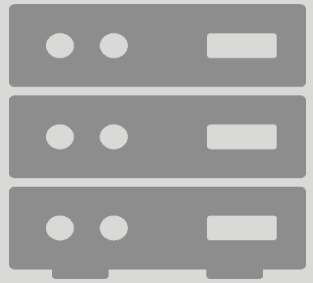
```
<!DOCTYPE html>
<html class="client-nojs" lang="ko" dir="ltr">
<head>
<meta charset="UTF-8"/>
<title>위키백과, 우리 모두의 백과사전</title>
<script>document.documentElement.className="client-js";RLCONF=
"pageLanguageDir":"ltr","pageVariantFallbacks":"ko"},"wgMFDisp
"skins.vector.user.styles":"ready","ext.gadget.SectionFont":"
"ext.centralNotice.startUp","ext.gadget.directcommons","ext.ga
<script>(RLQ=window.RLQ||[]).push(function(){mw.loader.implem
<link rel="stylesheet" href="/w/load.php?lang=ko&amp;modules=
<script async="" src="/w/load.php?lang=ko&amp;modules=startup&
<meta name="ResourceLoaderDynamicStyles" content=""/>
<link rel="stylesheet" href="/w/load.php?lang=ko&amp;modules=
<link rel="stylesheet" href="/w/load.php?lang=ko&amp;modules=
<meta name="generator" content="MediaWiki 1.40.0-wmf.10"/>
<meta name="referrer" content="origin"/>
<meta name="referrer" content="origin-when-crossorigin"/>
<meta name="referrer" content="origin-when-cross-origin"/>
<meta name="robots" content="max-image-preview:standard"/>
```

웹 통신

위키백과 메인 페이지 보고 싶어!

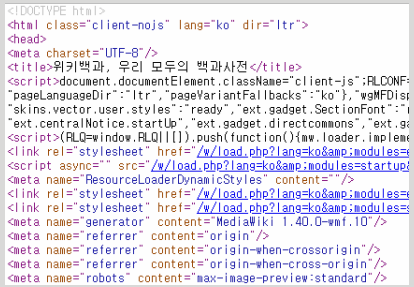
https://ko.wikipedia.org/

Client



Server

저장하고 있던
정적리소스 (HTML)



텍스트로 이루어진 문서가
이미지+디자인이 입혀진
형태로 보여지는 이유는?

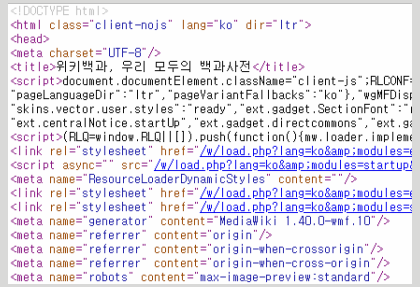
웹 통신

Client

렌더링

웹 브라우저

정적리소스 (HTML …)



Server

렌더링 : HTML, CSS, JavaScript로 작성된 문서를 파싱(텍스트 분해)하여 브라우저에 시각적으로 출력하는 것

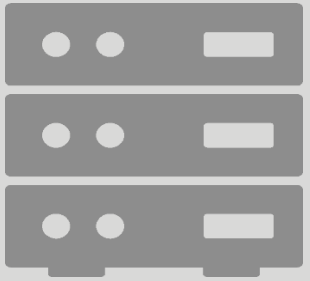
웹 통신

Client

네이버 메인 페이지보고 싶어!

https://www.naver.com/

사용자의 요청정보를
담고 있는 HTML 문서를
정적웹서버를 사용해 응답하려면?



Server

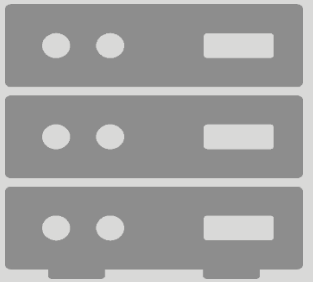
웹 통신

Client

네이버 메인 페이지보고 싶어!

https://www.naver.com/

사용자의 요청정보를
담고 있는 HTML 문서를
정적웹서버를 사용해 응답하려면?



Server

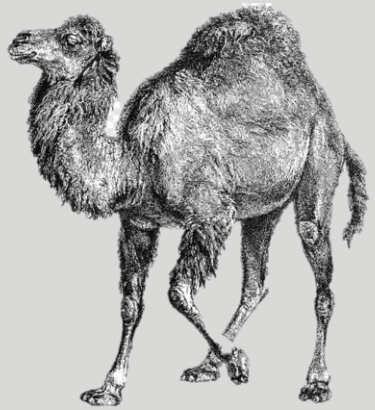
모든 경우의 수에 따른 HTML 문서를 만들어서 저장해놓고 있어야 함

Why? 정적웹서버는 문서 가공이 불가능하기 때문에!

웹 통신

클라이언트의 요청에 맞는 HTML 파일을
만들어주는 서버 프로그램이 필요하다!

서버 사이드 스크립트 언어



JS

php

Servlets & Jsp

Microsoft® .NET

Servlet

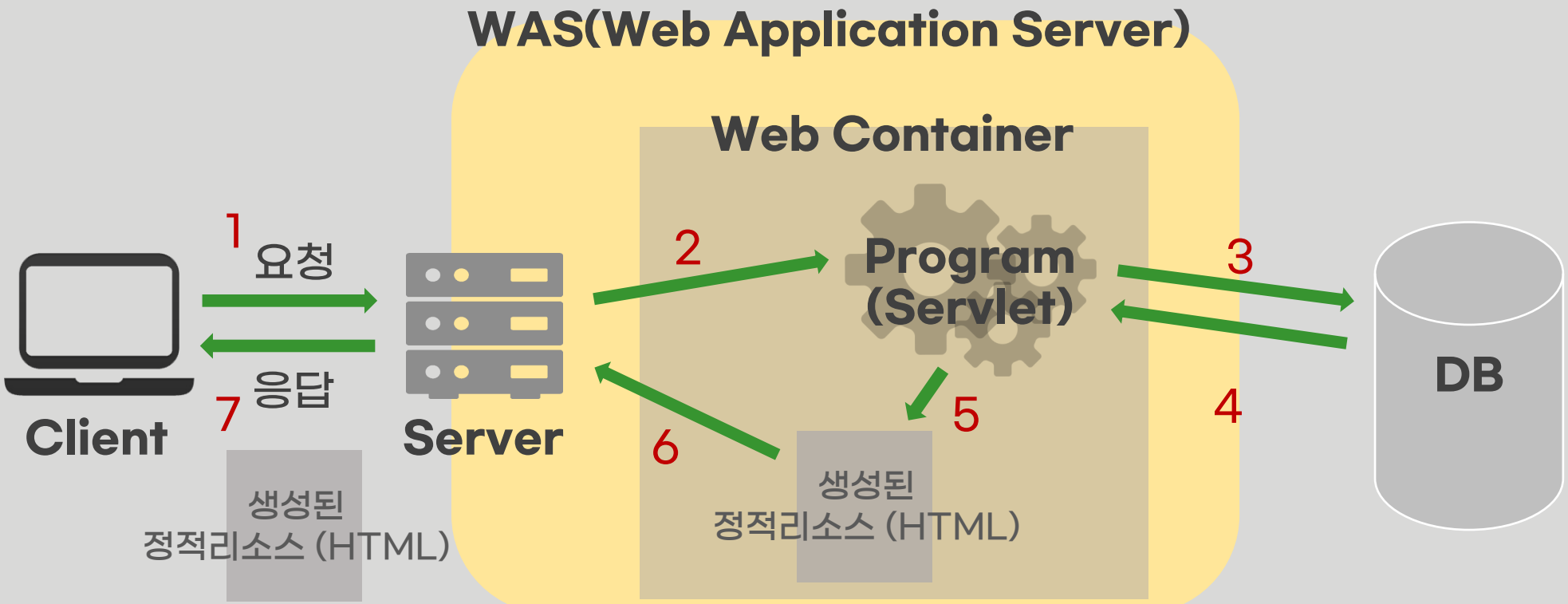
Servlet

= Server + Applet

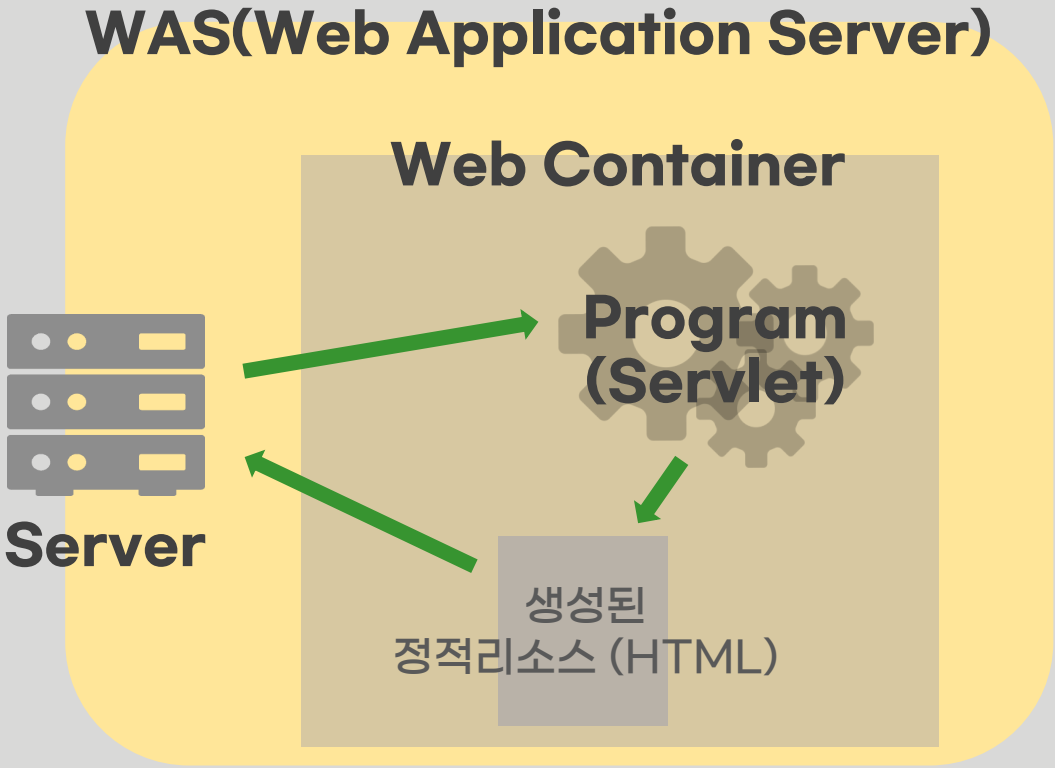
서버 상에서 작동하는 응용 프로그램

- Java를 기반으로 하는 Web Application Programming 기술
- Client 요청에 동적으로 작동, 응답은 가공이 완료된 정적문서 형식으로 제공

Servlet의 구조


WAS(Web Application Server)
Web Container
Client
Server
Program
(Servlet)
DB
1 요청
7 응답
2
3
4
5
6
생성된
정적리소스 (HTML)
생성된
정적리소스 (HTML)

Servlet의 구조



Web Server
: 정적인 컨텐츠를 제공하는 서버
: 요청을 컨테이너로 전달하고 결과를 넘겨주는 역할

WAS(웹 서버+웹컨테이너)
: 동적으로 컨텐츠를 생성하여 제공하는 서버

Web Container
: JSP와 Servlet을 실행시킬 수 있는 SW

Servlet의 특징

- .java 확장자를 가짐
- Java Multi Thread를 이용하여 동작함 -> 속도와 메모리 면에서 효율적임
- 객체지향적 -> 대규모 Application 개발에 적합함
- HttpServlet 클래스를 상속받음

Servlet의 특징

동적으로 리소스를 생성하여 제공하기 위한 방법

1) CGI(Common Gateway Interface)

2) WAS(Web Application Server)

프로그램 / 프로세스 / 스레드

프로그램(Program) :

컴퓨터에서 어떤 작업을 위해 실행할 수 있는 '정적인 상태'의 파일
ex) Windows의 exe파일

프로세스(Process) :

프로그램이 실행되서 돌아가고 있는 상태, 컴퓨터에서 연속적으로 실행되고 있는
'동적인 상태'의 컴퓨터 프로그램, 운영체제가 메모리 등의 필요한 자원을 할당받은 상태

스레드(Thread) :

프로세스가 할당 받은 자원을 이용하는 실행 단위
프로세스의 특정한 수행 경로이자 프로세스 내에서 실행되는 여러 흐름의 단위

프로그램 / 프로세스 / 스레드

운영체제로부터
자원을 할당받은 상태

Program

Process

thread

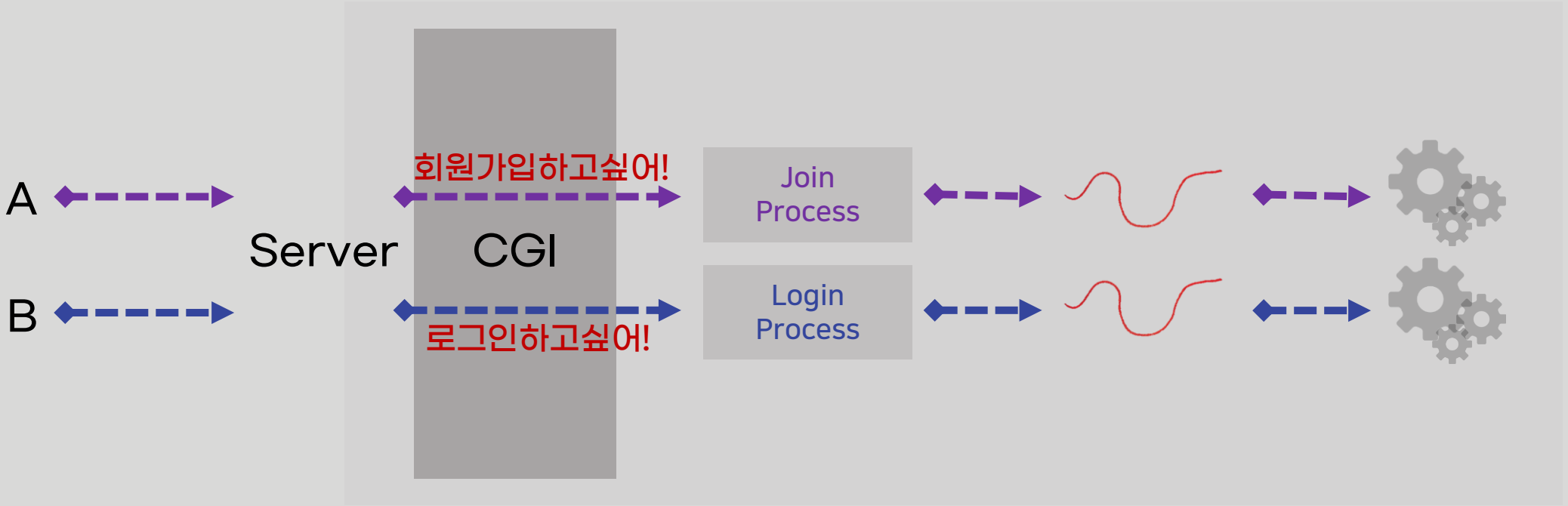
Servlet의 특징

동적으로 컨텐츠를 제공하기 위한 방법

1) CGI(Common Gateway Interface)

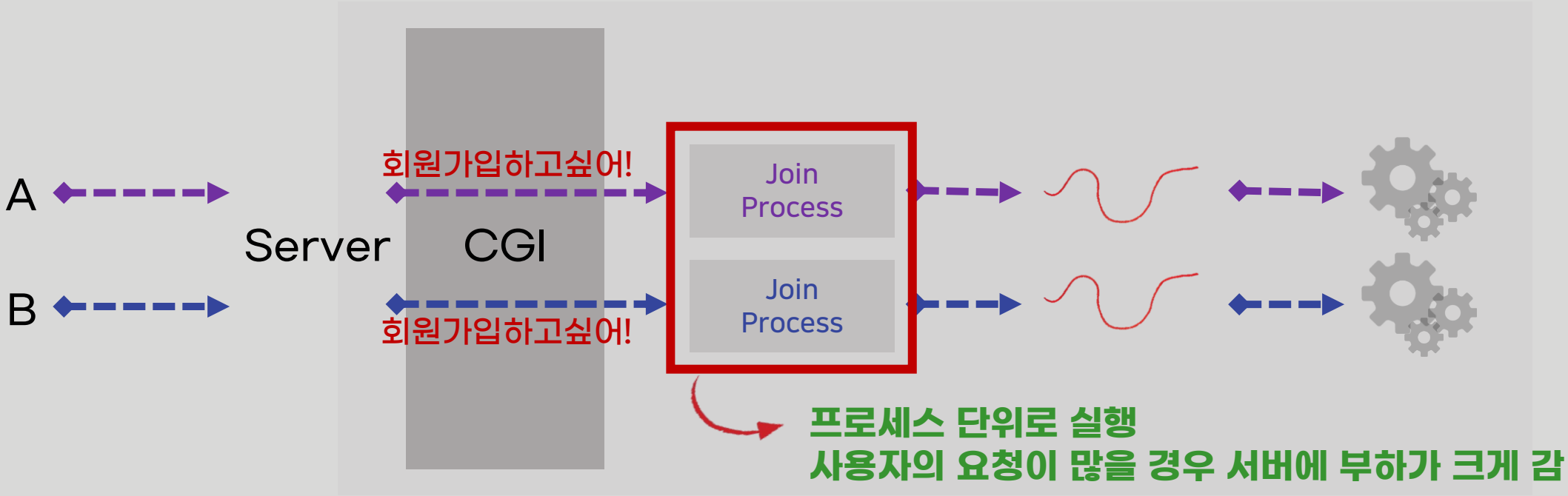
Servlet의 특징

1) CGI(Common Gateway Interface)



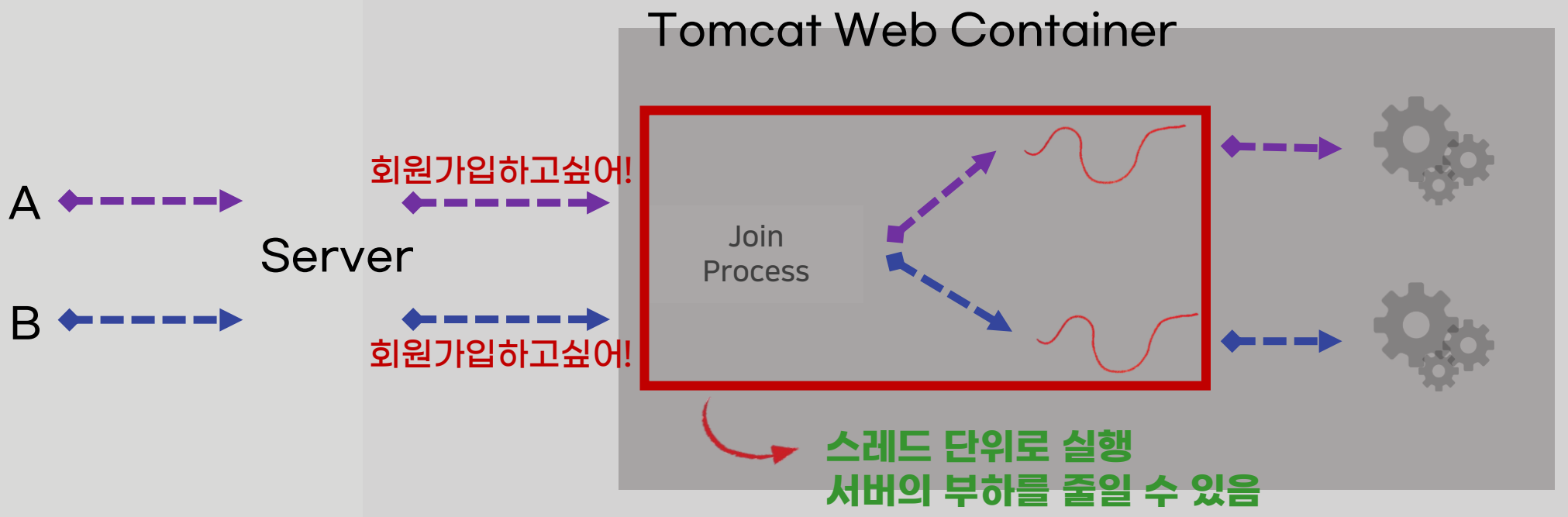
Servlet의 특징

1) CGI(Common Gateway Interface)



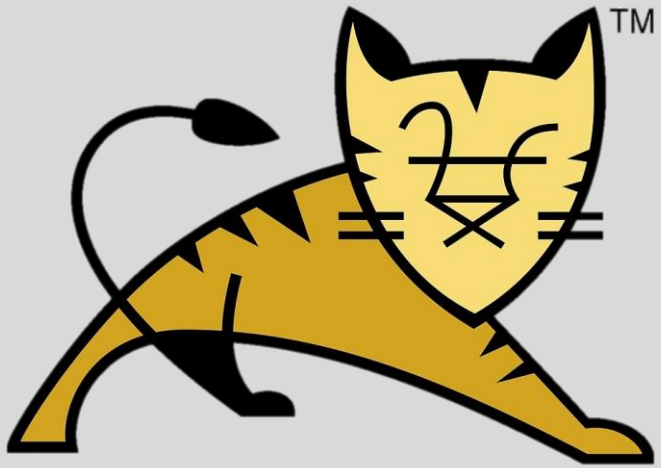
Servlet의 특징

2) WAS(Tomcat) - Servlet



Tomcat 설치하기

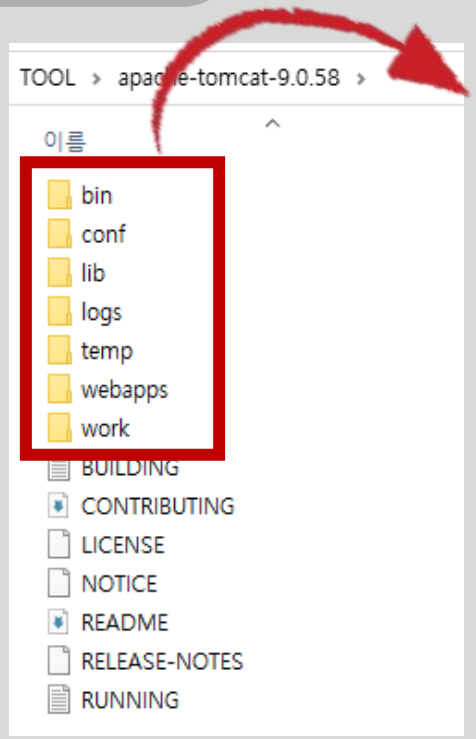
step1 **Tomcat zip 다운로드 받기**



https://tomcat.apache.org/download-90.cgi

Tomcat 설치하기

step2 **zip 파일 압축풀기**



bin : 톰캣을 실행하고, 종료시키는 스크립트(.bat, .sh 등) 파일

conf : 서버 전체 설정파일 폴더 (server.xml 등)

lib : 톰캣 구동하는데 필요한 라이브러리(jar)

logs : 예외 발생 사항 등의 로그 저장

temp : 임시 저장용 폴더

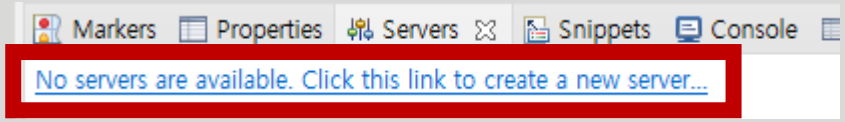
webapps : 웹 어플리케이션 폴더

work : jsp 파일을 서블릿형태로 변환한 java 파일과 class 파일이 저장

Eclipse에서 Tomcat 연동하기

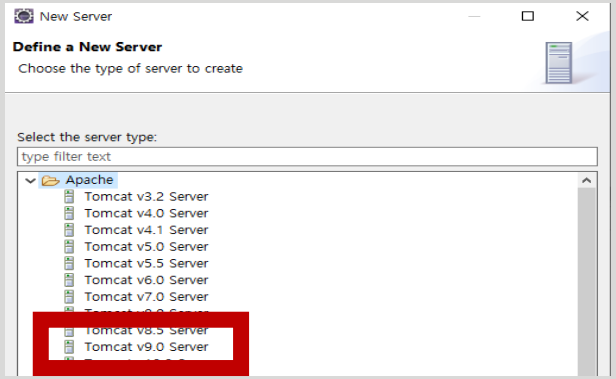
step1

Server 탭에서 새 서버 연결 링크 클릭하기



step2

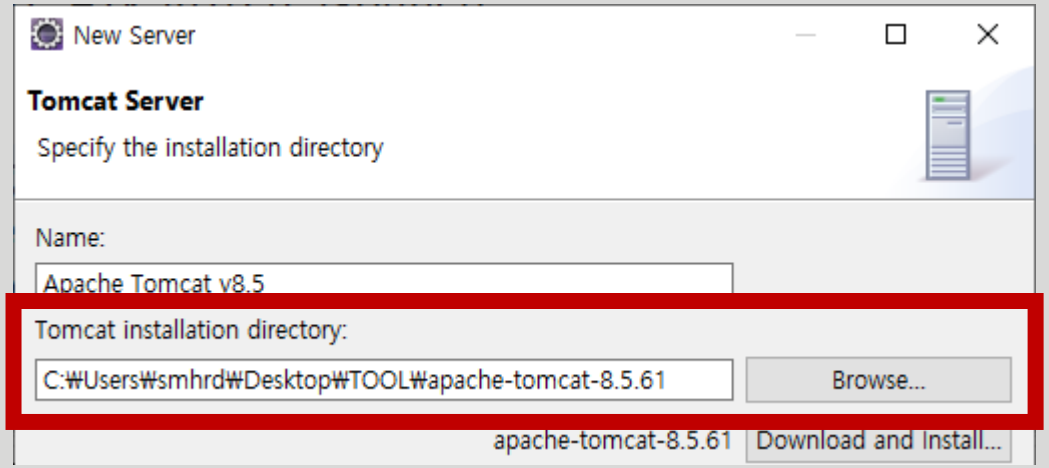
New Server 창에서 Apache-연결할 서버 버전 선택



Eclipse에서 Tomcat 연동하기

step3

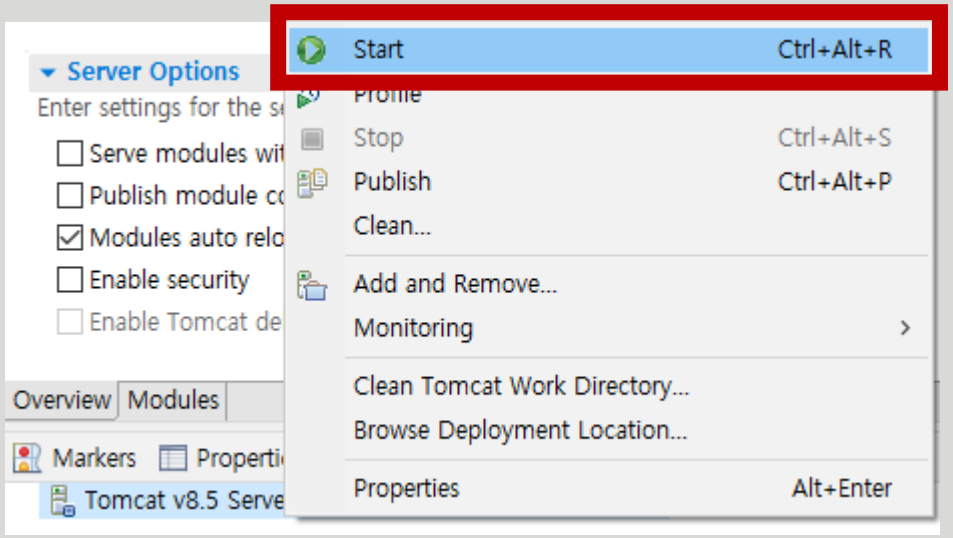
Tomcat 폴더 선택



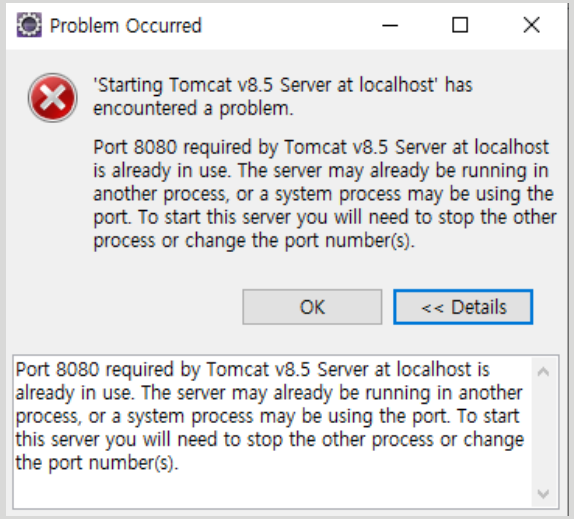
Eclipse에서 Tomcat 연동하기

step4

Tomcat Server 시작하기



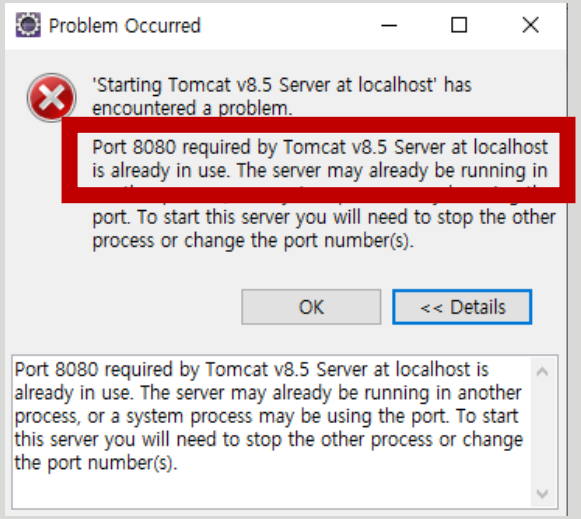
오류 발생!



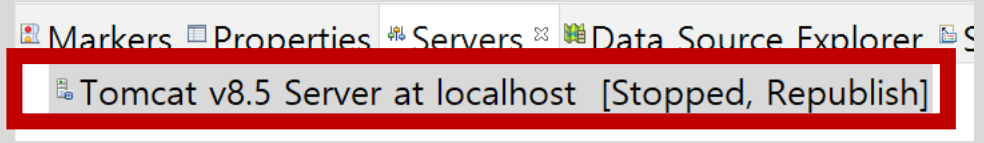
Eclipse에서 Tomcat 연동하기

step5

'Port 8080 required by Tomcat v8.5 …' 문제 해결하기



8080 Port가 이미 다른 용도로 사용 중이라 (Oracle)
Tomcat Server에서 사용할 수 없음

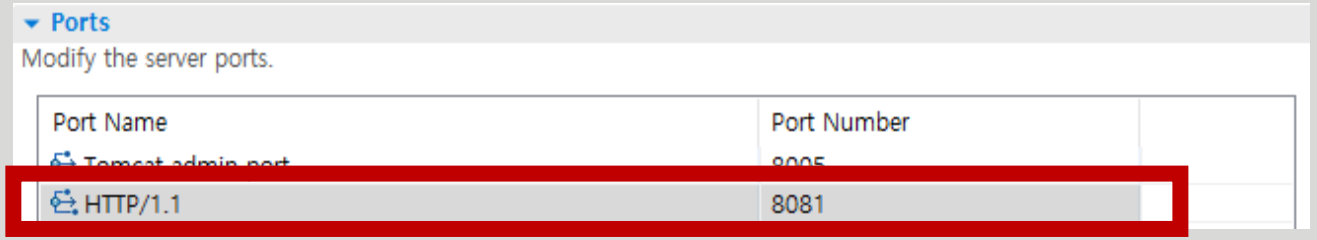


더블 클릭 -> Tomcat v8.5 Server at localhost 창이 나타남

Eclipse에서 Tomcat 연동하기

step5

‘Port 8080 required by Tomcat v8.5 …’ 문제 해결하기



8080 이 아닌 번호로 HTTP Port Number 수정
(이미 사용중인 Port Number 가 있을 수 있으니 주의)

문자 인코딩 설정

문자 인코딩(Character Encoding)이란?

사용자가 입력한 문자나 기호들을 컴퓨터가 이용할 수 있는 신호로 만드는 것

* 디코딩(Decoding) : 0과 1로 구성된 바이너리 데이터를 다시 문자로 복구하는 것

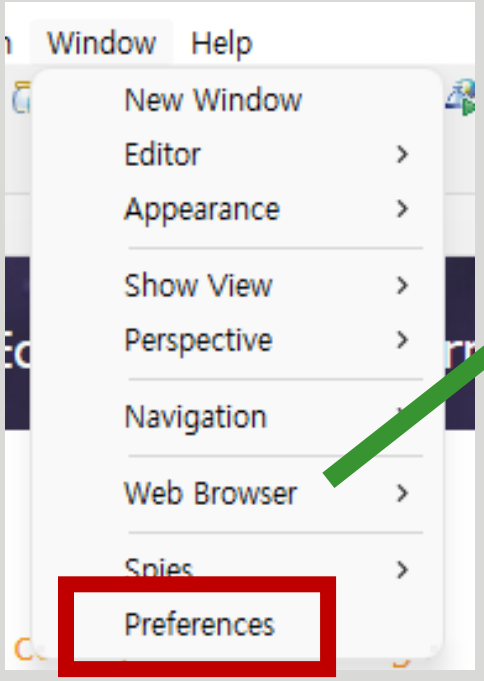
UTF-8 : [조합형] 전세계의 모든 문자를 한꺼번에 표현할 수 있는 인코딩

EUC-KR : [완성형] 한글/한국에서 통용되는 한자/영문 표현할 수 있는 인코딩

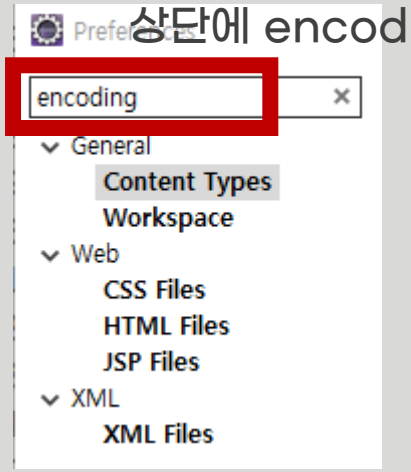
: 문자표에 없는 문자는 표현 불가능

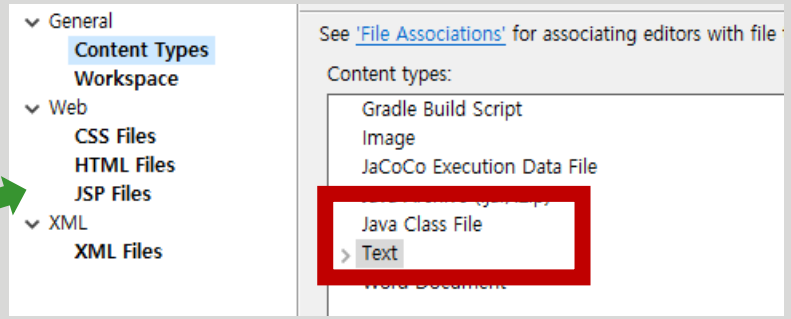
문자 인코딩 설정

* Eclipse에서 문자 인코딩 설정하는 방법

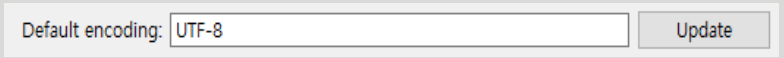


상단에 encoding 검색





Content-Types에 Java Class File과 Text의 Default encoding을 UTF-8로 작성

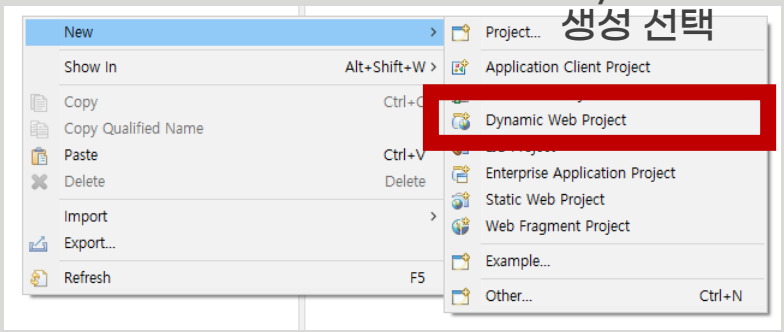


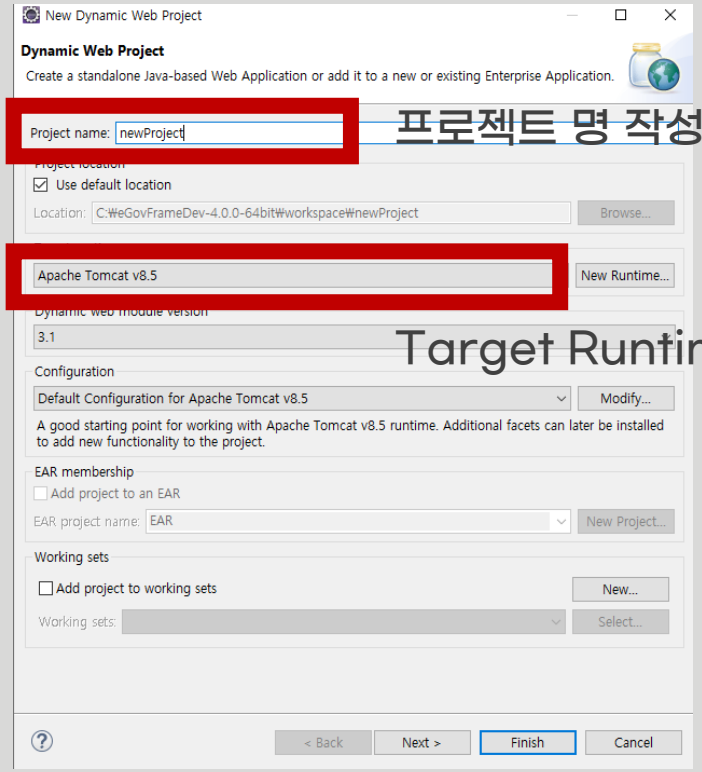
나머지 메뉴들은 모든 Encoding을 UTF-8로 선택

Dynamic Web Project 생성하기

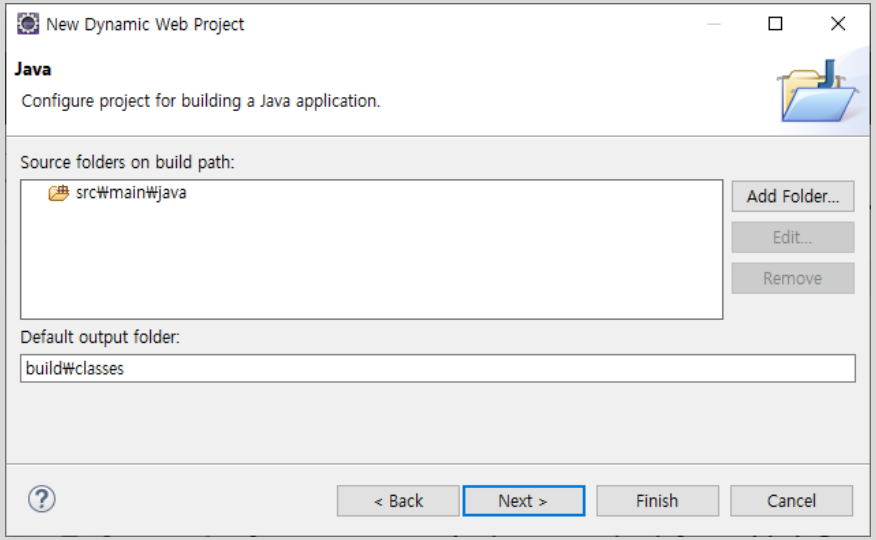
* Eclipse에서 Dynamic Web Project 생성하기

Dynamic Web Project 생성 선택

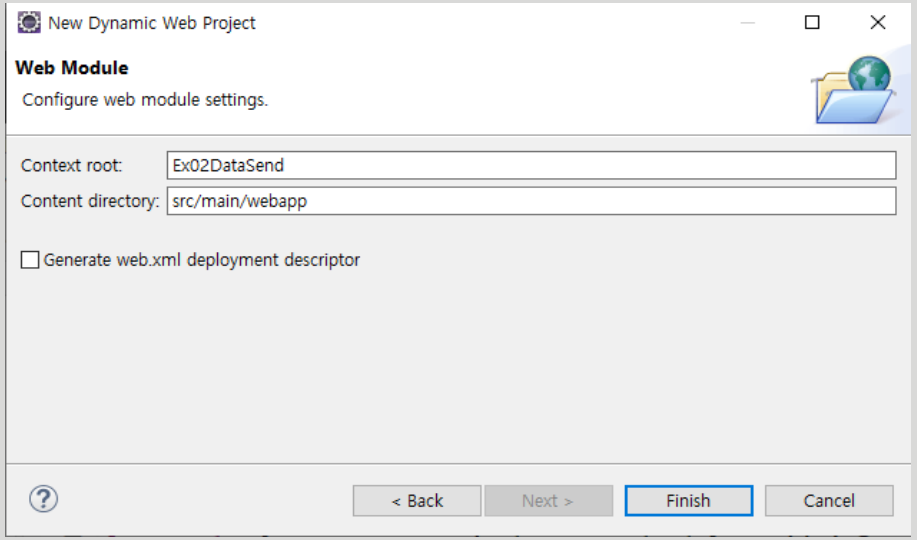




Dynamic Web Project 생성하기

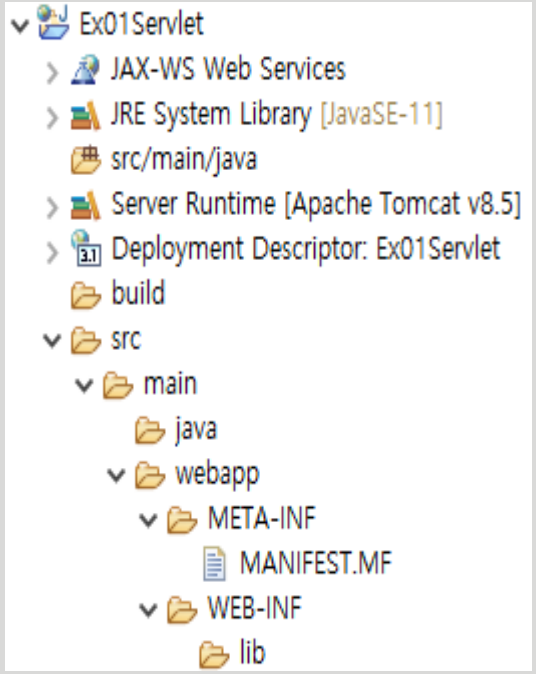


. java Source 파일 경로 설정



Context root : 웹 어플리케이션 이름(식별자), URL에서 사용
Content directory : 웹 콘텐츠 파일 저장할 작업 폴더 이름

Dynamic Web Project 생성하기



src : Java 소스파일, 프로퍼티(.properties) 파일이 위치하는 폴더

build : Java 클래스(.class)파일이 위치하는 폴더

webapp : HTML, CSS, JS, JSP, 이미지 파일들의 웹 콘텐츠가 위치하는 폴더

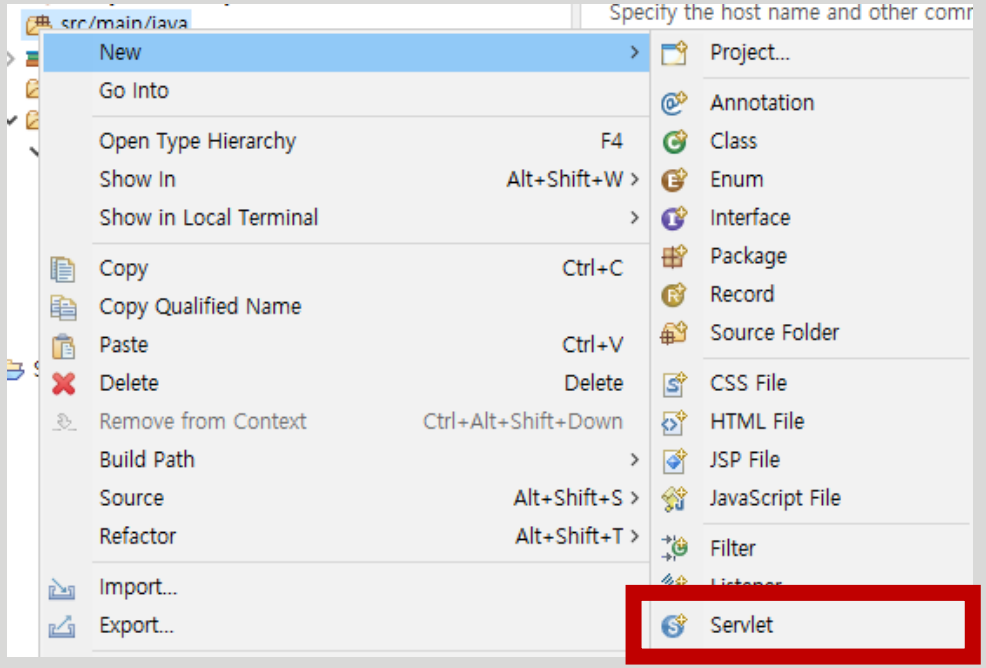
WEB-INF : 웹 어플리케이션 설정 관련 파일들이 위치하는 폴더
이 폴더에 있는 파일은 클라이언트에서 요청할 수 없음

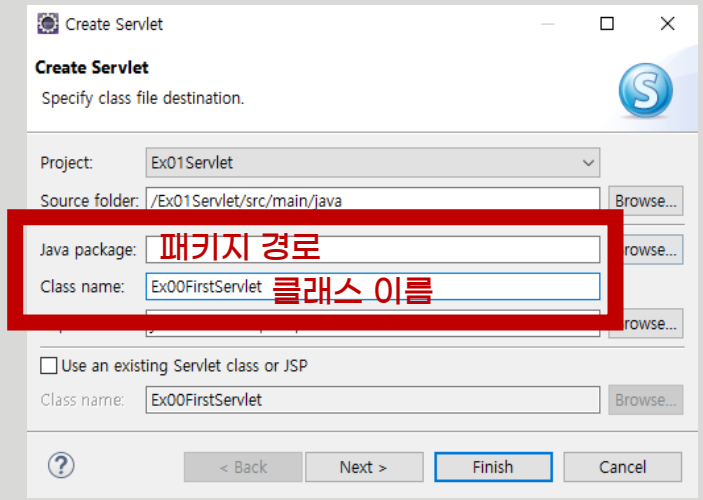
lib : 자바 아카이브 파일(.jar)이 위치하는 폴더

META-INF : 일종의 사용설명서인 MANIFEST를 담기위한 폴더

Servlet 만들기

Step1 Servlet Class 이름 작성하기 (절대 바로 Finish 누르지 않고 Next 누르기)





Servlet 만들기

Step2 **URL-Mapping 값 정하기**

URL Mapping :

Web browse에서 Servlet을 동작시키기 위해 실제 Java 클래스의 이름 대신,
Servlet을 요청하기 위한 문자열을 Servlet 클래스와 Mapping(매핑) 시키는 것

원래주소 : http://localhost:8081/FirstProject/Servlet/HelloWeb

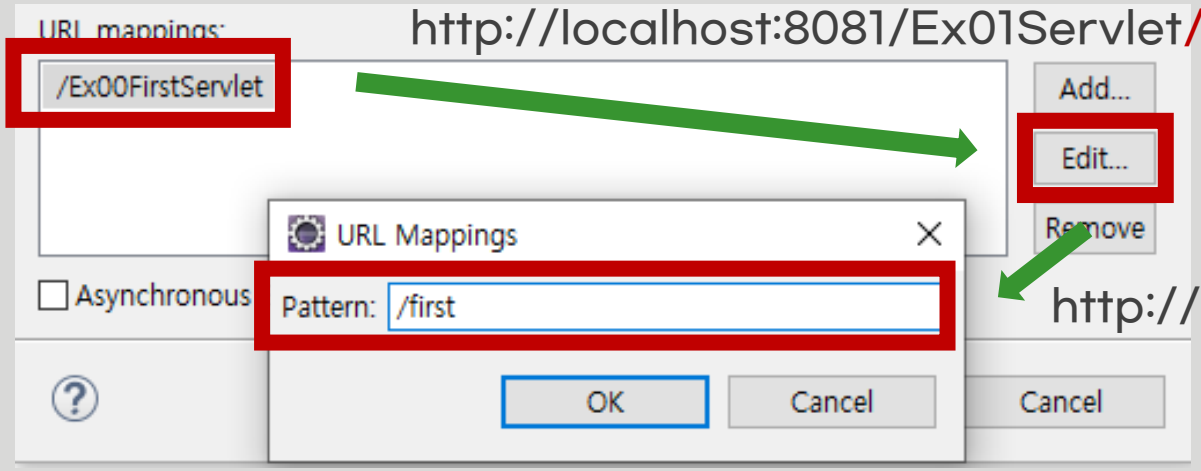
주소가 길다 -> 불편함
경로가 드러난다 -> 보안상 문제

매핑된 주소 : http://localhost:8081/FirstProject/hweb

Servlet 만들기

Step2 URL-Mapping 값 정하기

기본값 : class 명과 동일



http://localhost:8081/Ex01Servlet/Ex00FirstServlet

http://localhost:8081/Ex01Servlet/first

Servlet 만들기

Step2 URL-Mapping 값 정하기

URL Mapping 방법 1. annotation 사용

컴파일이나 배포, 실행할 때 참조할 수 있는 주석
클래스나 필드, 메서드에 대해 부가정보를 등록할 수 있음
프로그램의 의미적인 부분에 직접 영향을 주지 않음

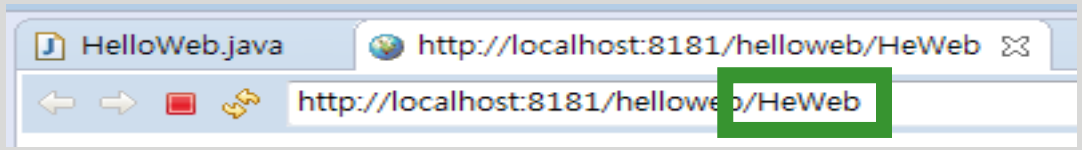
```java
package com.javalec.ex;

import java.io.IOException;

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

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

@WebServlet("/매핑한문자열")



Servlet 만들기

Step2 URL-Mapping 값 정하기

URL Mapping 방법 2. web.xml 사용

웹 어플리케이션을 실행시킬 때 필요한 설정 정의

```
<servlet>
    <servlet-name>helloworld</servlet-name>
    <servlet-class>com.servlet.helloworld</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>helloworld</servlet-name>
    <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

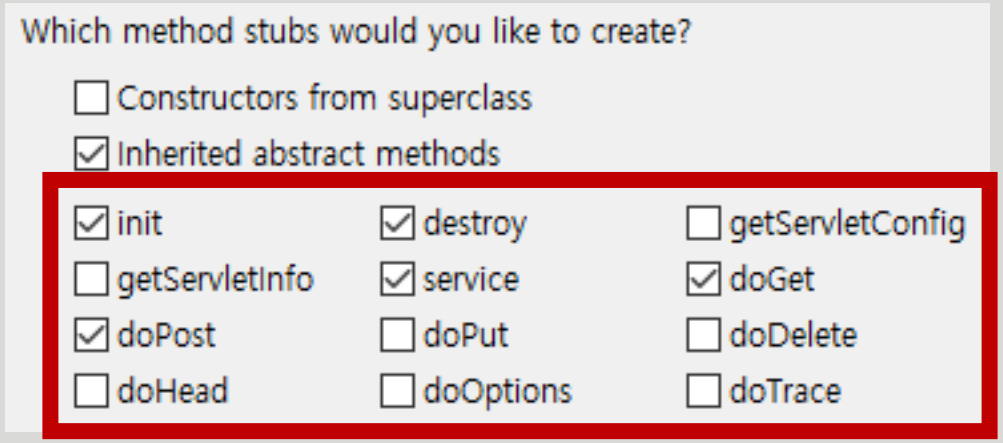
servlet-name : servlet 별명

servlet-class : servlet class의 실제 경로

url-pattern : URL 매핑 값 작성

Servlet 만들기

Step3 **상속 메서드 선택**



상위 클래스에 정의되어 있는
추상 메서드 상속 허용(재정의)

* init, destroy, service, doGet, doPost는 서블릿 생명 주기(Life Cycle)에 해당

Servlet 만들기

Step3 **상속 메서드 선택**

서블릿 생명주기(Life Cycle)

: 서블릿 객체 생성부터 소멸까지의 과정

Servlet 만들기

Step3 **상속 메서드 선택**

1) 클라이언트로 부터 최초 요청이 들어오면 Servlet 클래스가 로딩되어 요청에 대한 Servlet 객체 생성

생성자 호출 -> Servlet 으로서의 역할을 하진 못함

init()

2) Servlet 초기화 (Servlet으로서의 역할 수행)

비용이 많은 작업 이므로 서블릿이 처음 생성되었을 때 딱 한번만 호출 (자원절약)

Get요청시

service()

doGet()

doPost()

3) HTTP 요청 메서드에 따라서 doGet() or doPost() 호출

서블릿 객체가 생성되어 있을 경우에는 service만 호출

Post요청시

destroy()

4) Servlet 객체 소멸 (ex. 서버 종료, 재시작 등)

서버 종료 시 딱 한번만 호출

Servlet 만들기

URL-Mapping Annotation

HttpServlet 상속

```
@WebServlet("/first")
public class Ex00FirstServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

모든 데이터는 바이트 형태로 전송

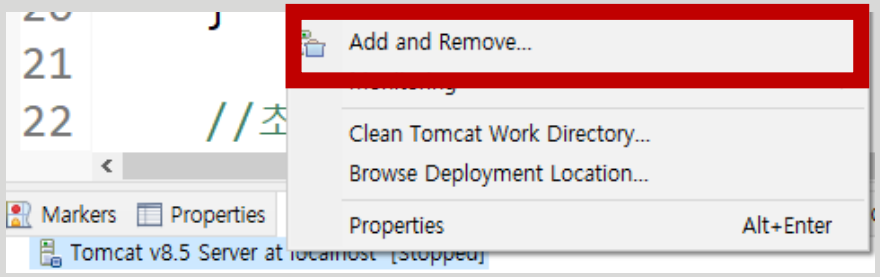
Servlet 객체도 바이트 형태로 전송되어야 함

자바 객체를 바이트 형태로 변환(직렬화) 하는 과정(Serialization)이 필요

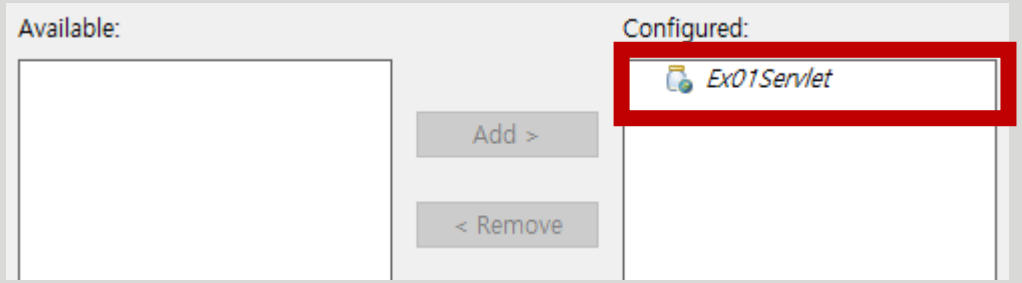
다시 바이트 형태의 데이터를 자바 객체로 변환(역직렬화) 하는 과정(Deserialization) 또한 필요

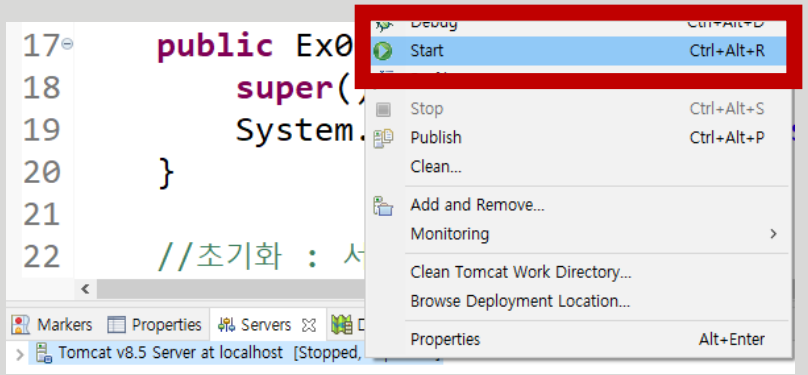
-> 이 때 클래스 버전을 확인하기 위해 필요한 것이 serialVersionUID

Servlet 실행하기



톰캣 서버에 프로젝트 올리기





톰캣 서버 시작!

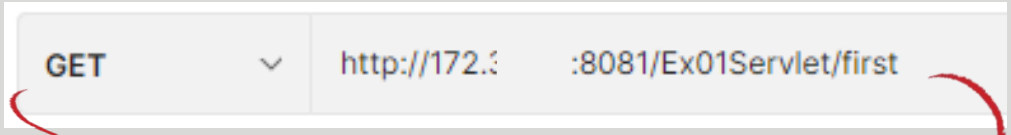
POSTMAN 활용해서 요청해보기

개발자들이 API를 디자인하고 빌드하고
테스트하고 반복하기 위한 API 플랫폼

* API(Application Programming Interface)
: 응용 프로그램에서 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스

POSTMAN 활용해서 요청해보기

Step1 **New request 생성 후 HTTP 요청 메서드, 요청 URL 작성**



GET 방식으로 요청

http://172.00.0.0:8081/Ex01Servlet/first

- http: 클라이언트-서버 프로토콜
- 172.00.0.0 (서버 pc ip주소 (localhost : 127.0.0.1)) → Domain(도메인 : 네트워크 호스트 이름) ex. Naver.com
- 8081 (서버 포트번호)
- Ex01Servlet (Context path) → 웹 어플리케이션을 구분하기 위한 경로 (절대로 다른 프로젝트와 겹치면 안됨)
- first: URL 매핑 값

POSTMAN 활용해서 요청해보기

Step1 **New request 생성 후 HTTP 요청 메서드, 요청 URL 작성**

* 내 컴퓨터 ip 주소 찾는 방법

```
C:₩Users₩예딘>ipconfig

Windows IP 구성

이더넷 어댑터 이더넷:

   연결별 DNS 접미사. . . . :
   링크-로컬 IPv6 주소 . . . . : fe80::848b:c495:4afe:171f%4
   IPv4 주소 . . . . . . . . . : 172.30.1.2
   서브넷 마스크 . . . . . . . : 255.255.255.0
   기본 게이트웨이 . . . . . . : 172.30.1.254
```

IPv4 : 최초의 안정적인 인터넷 프로토콜(IP)

IPv6 : IPv4의 확장성과 용량 면에서의 한계를 보완하기 위해 나온 IP

서브넷 마스크 : IP주소를 네트워크 및 호스트 주소로 분리 시 사용

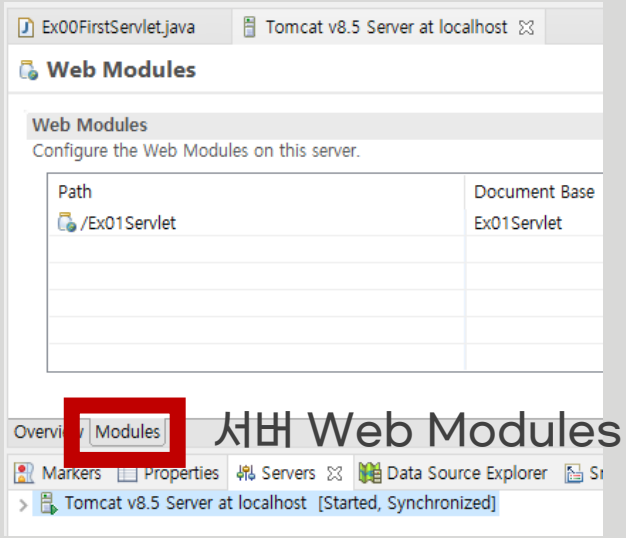
기본 게이트웨이 : 라우터(공유기 등)의 주소

* 라우터 : 네트워크와 네트워크를 연결해주는 장비

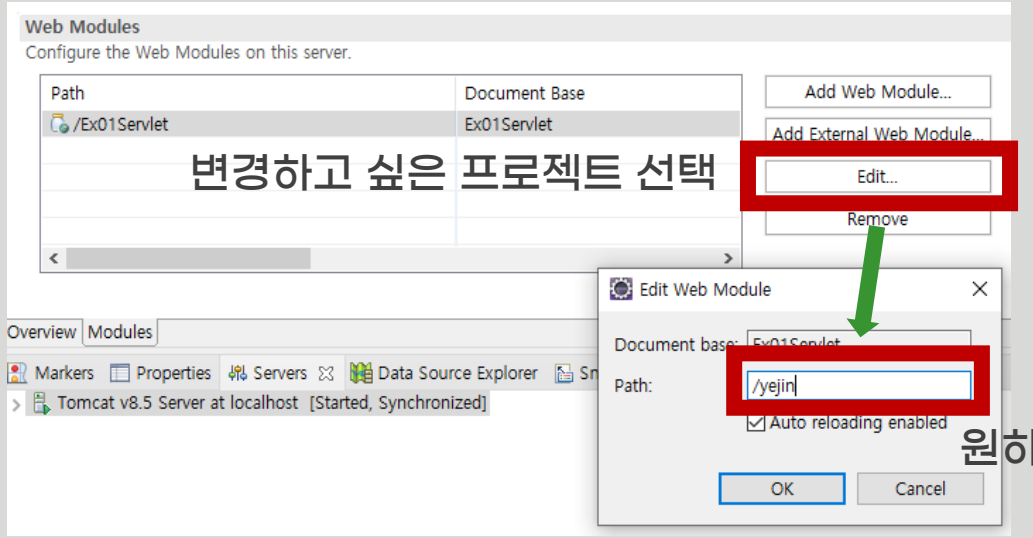
POSTMAN 활용해서 요청해보기

Step1 **New request 생성 후 HTTP 요청 메서드, 요청 URL 작성**

* Context-path 수정하는 방법



서버 Web Modules 확인



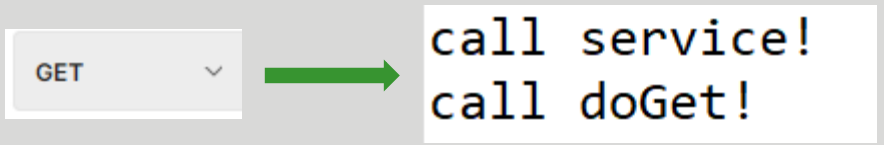
원하는 path로 수정

POSTMAN 활용해서 요청해보기

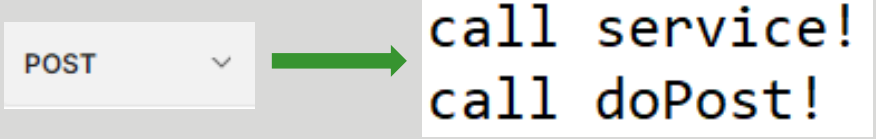
Step2 **요청 전송 후 eclipse 콘솔창에서 서블릿 생명주기 확인**

```
INFO: Server startup in 383 ms
call constructor!
call init!
call service!
call doGet!
12월 12, 2022 10:44:01 오전 org.a
INFO: 셧다운 포트를 통해 유효한 셧다운
12월 12, 2022 10:44:01 오전 org.a
INFO: 프로토콜 핸들러 ["http-nio-80
12월 12, 2022 10:44:01 오전 org.a
INFO: 서비스 [Catalina]을(를) 중지
WARNING: An illegal reflective
WARNING: Illegal reflective acc
WARNING: Please consider report
WARNING: Use --illegal-access=w
WARNING: All illegal access ope
call destroy!
```

* Get 요청 시



* Post 요청 시



POSTMAN 활용해서 요청해보기

Step3 **url-mapping 값 수정 후 결과 확인하기**

```
<servlet>
    <servlet-name>first</servlet-name>
    <servlet-class>Ex00FirstServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>first</servlet-name>
    <url-pattern>/example</url-pattern>
</servlet-mapping>
```

http://172.3 :8081/yeji /example →

```
call service!
call doGet!
```

request와 response

```
protected void service(HttpServletRequest request, HttpServletResponse response)
```

HttpServletRequest

: 요청 받을 때 전달받은 정보를 저장하고 서블릿에게 전달하기 위한 목적으로 사용

HttpServletResponse

: 응답할 때 전달할 정보를 저장하고 클라이언트에 돌려주기 위한 목적으로 사용

request와 response

네이버에 로그인 하고싶어!

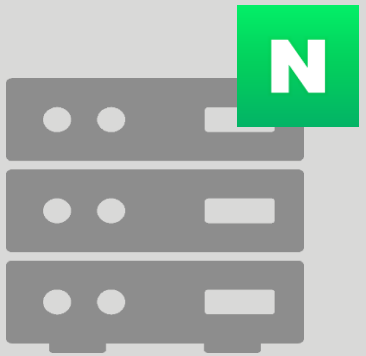
클라이언트가 서버에게 요청

Client

Request

(HttpServletRequest)

(header정보, 파라미터, Cookie,
URL, Body Stream 등)



Server

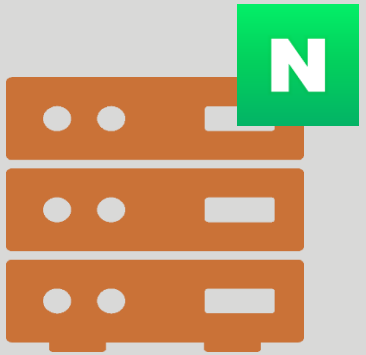
request와 response

사용자의 정보가 담긴
네이버 메인페이지를 보여줄래!

서버가 클라이언트에게 응답

Client

Response
(HttpServletResponse)
(콘텐츠 타입, 응답코드, 메시지 등)



Server

Servlet 만들기 실습1

Ex1 화면에 “Hello Servlet” 문자열을 출력하시오.

Source Ex01Print.java

```
PrintWriter out = response.getWriter();
```
텍스트 출력 스트림(통로) 생성

```
out.print("Hello Servlet!");
```
스트림(통로) 통해 텍스트 출력

Servlet 만들기 실습2

Ex2 화면에 굵은 글씨의 “Hello Servlet” 문자열을 출력하시오.

Hello Servlet!

Source Ex02PrintHtml.java

```
PrintWriter out = response.getWriter();

out.print("<b>Hello Servlet!</b>");
```

브라우저 상에서 는 태그로 인식

next

ch2. 데이터 전송

강예진 연구원