Servlets & Jsp

Web

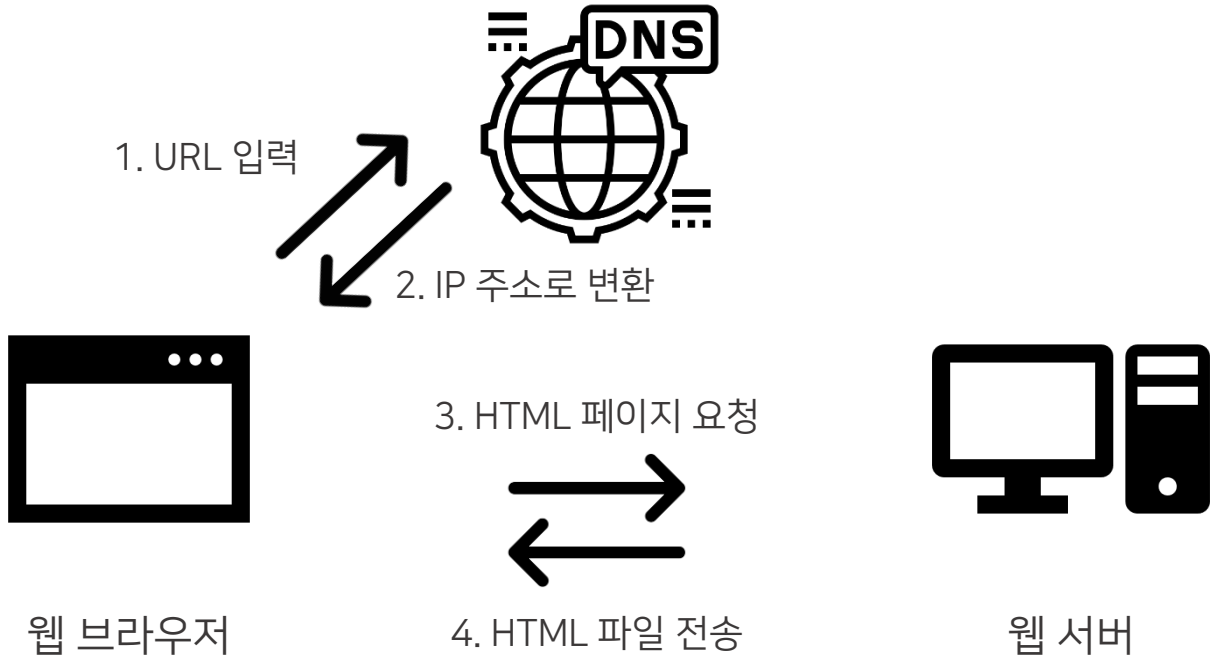
- 월드 와이드 웹(WWW)를 줄여서 웹이라고 한다.
- 웹은 인터넷에서 운영되는 서비스 중 하나로, 웹 자체가 인터넷을 의미하지는 않는다.
  하지만 대다수가 '웹 = 인터넷'이라 생각할 만큼, 그만큼 웹이 인터넷의 대표적인 서비스이다.

- 엄밀히 따지면,인터넷은 컴퓨터 네트워크망을 의미하고, 웹은 인터넷 서비스 중 하나를 의미한다.
- 웹은 인터넷 상의 정보를 하이퍼텍스트 방식과 멀티미디어 환경에서 검색할 수 있게 해주는 정보 검색 시스템을 말한다.
- 웹은 HTTP라는 프로토콜을 사용하며, HTML로 작성된 문서를 연결하여 다양한 콘텐츠를 제공한다.

Web의 동작 구조

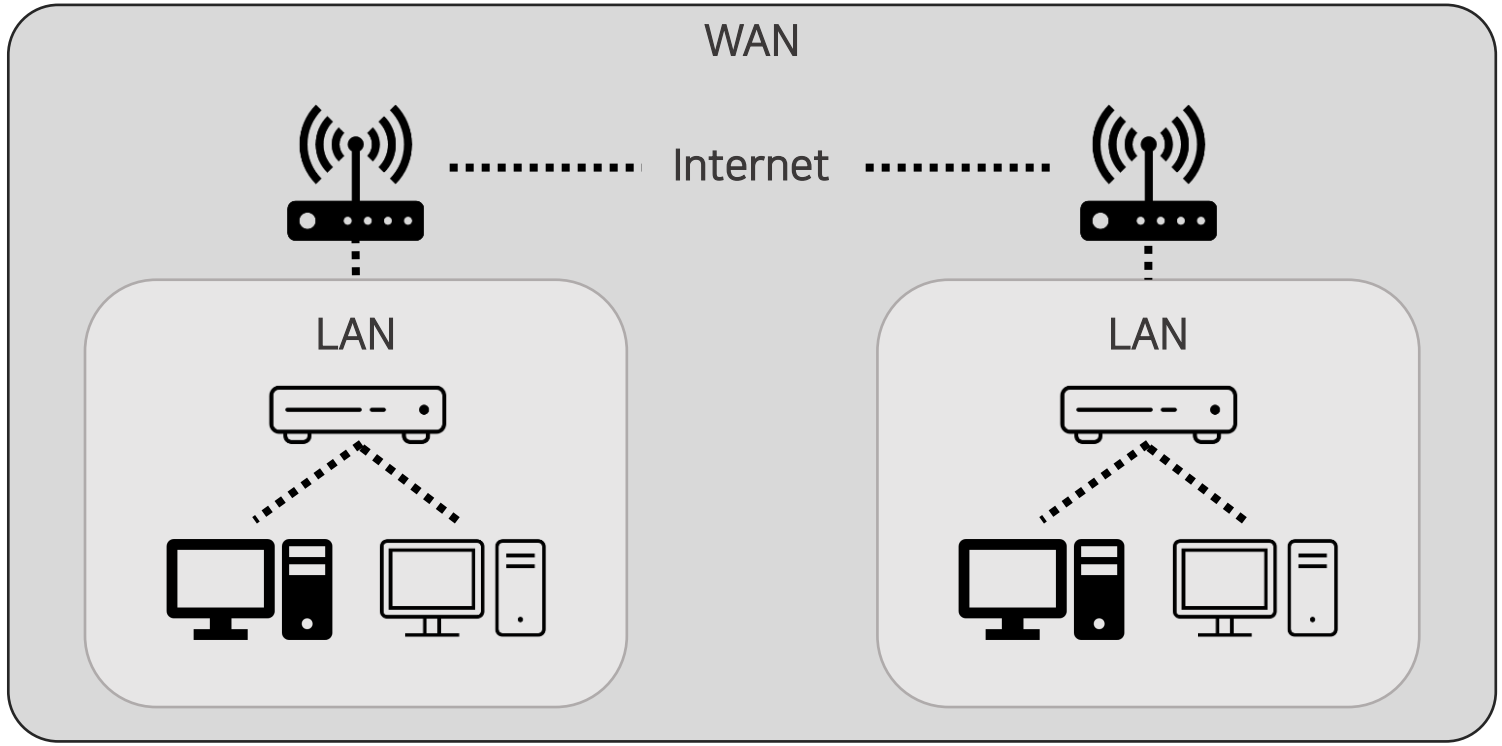
- HTML로 만들어진 콘텐츠는 웹 서버(Web Server)라고 하는 별도의 서버 소프트웨어를 통해 서비스로 제공된다.
- 웹 서버 소프트웨어로는 아파치(Apache)가 대표적이다.

- 또한 웹 서비스를 요청한 클라이언트에서 서비스를 이용하기 위해서는 웹 브라우저(Web Browser)가 필요하다.



네트워크

- 네트워크(Network)는 여러 컴퓨터들을 통신 회선으로 연결한 것을 말한다.
- LAN (Local Area Network)은 가정, 회사, 건물, 특정 영역에 존재하는 컴퓨터를 연결한 것이고,
- WAN(Wide Area Network)는 LAN을 연결한 것으로, 우리가 흔히 말하는 인터넷은 WAN이다.



네트워크

- 웹은 인터넷 기반 서비스 중 하나이고, 인터넷은 전 세계를 연결한 컴퓨터 네트워크이다.

- 네트워크(Network)는 컴퓨터와 컴퓨터를 연결해주는 망(Net)으로,
  네트워크를 구축하기 위해서는 컴퓨터 간의 연결 규격, 규약이 필요하다.
- 컴퓨터 간의 연결 규격, 규약을 프로토콜(Protocol)이라 한다.

프로토콜

- 컴퓨터 간의 통신을 할 수 있도록 만든 프로토콜에는 TCP 통신과 UDP 통신이 있다.

- TCP(Transmission Control Protocol)은 연결형 프로토콜로, 상대방이 연결된 상태에서 데이터를 주고 받는다.
  즉, 클라이언트의 연결 요청과 서버의 연결 수락으로 통신 회선이 고정되어, 데이터가 손실없이 순서대로 전달된다.
- TCP는 IP와 함께 사용하기 때문에 TCP/IP라고 한다.
  웹 브라우저가 웹 서버에 연결할 때, 이메일 전송, 파일 전송, DB 연동 등에 쓰인다.

- UDP(User Datagram Protocol)은 발신자가 일방적으로 수신자에게 데이터를 보내는 방식으로,
  TCP/IP 처럼 연결 요청 및 수락 과정이 없기 때문에 TCP보다 데이터 전송 속도가 상대적으로 빠르다.
- 하지만 고정 회선이 아니라 여러 회선을 통해 데이터가 전송되기 때문에
  특정 회선의 속도에 따라 데이터가 순서대로 전달되지 않거나, 잘못된 회선으로 인해 데이터 손실이 발생할 수 있다.

TCP/IP

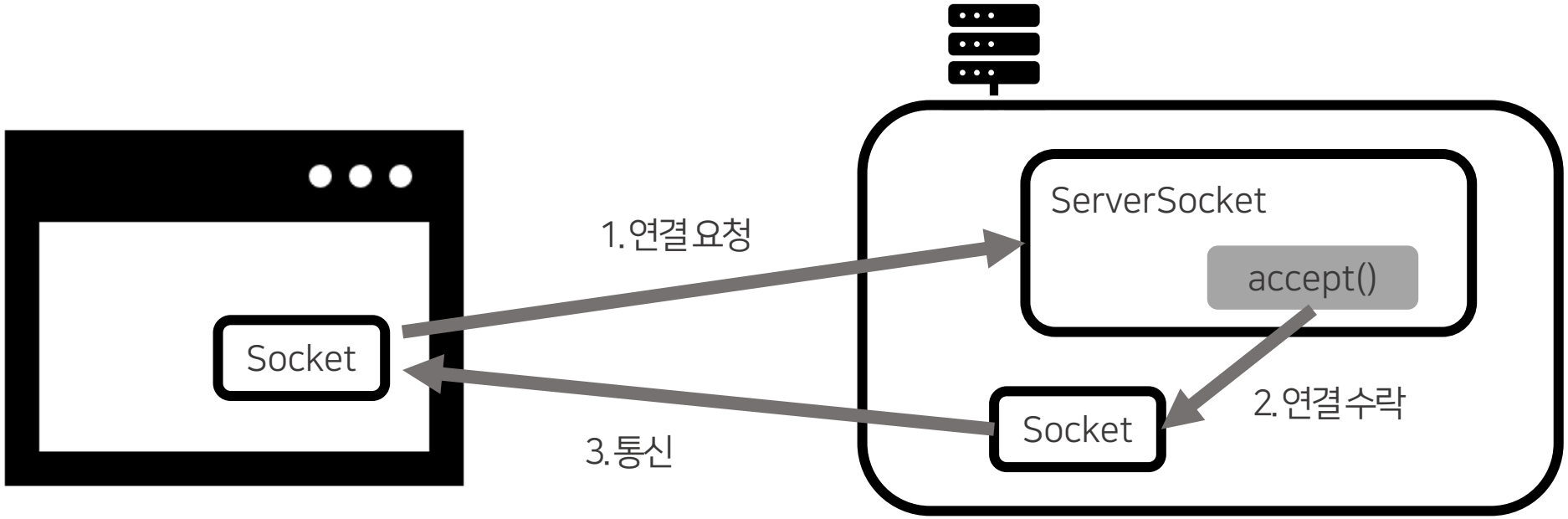
- 개방형 구조를 갖는 TCP/IP는 하드웨어, 운영체제, 접속 매체와 관계없이 동작할 수 있다.

- TCP/IP는 OSI(Open Systems Interconnection) 7계층에서 유래한 것으로 4계층 구조로 구성되어 있다.
  - 응용 계층 - WWW, FTP, Telnet, SMTP와 같은 네트워크 프로그램
  - 전송 계층 - 각 시스템을 연결하고 TCP 프로토콜을 이용해 데이터를 전송
  - 인터넷 계층 - IP 프로토콜을 이용해 데이터를 정의하고 경로를 배정
  - 물리 계층 - 실제 네트워크에 접근할 수 있도록 하는 물리적인 부분

- OSI 7계층은 네트워킹을 위한 물리적 장비에서 실 서비스를 제공하기 위한 애플리케이션에 이르는 단계까지 계층화 한 것이다.
- TCP/IP는 이러한 OSI 7계층을 좀 더 단순화하여 4계층으로 정의한 것이다.

TCP 네트워킹

- Java는 TCP 네트워킹을 위해 java.net의 ServerSocket 클래스와 Socket 클래스를 제공한다.
- ServerSocket은 클라이언트의 연결을 수락하는 서버 쪽 클래스이고,
  Socket은 클라이언트에서 연결을 요청할 때 또는 클라이언트와 서버 양쪽에서 데이터를 주고받을 때 사용되는 클래스이다.



IP 주소

- 각각의 집마다 고유한 주소가 있기 때문에 우편물이나 택배물이 정확하게 집을 찾아 도착할 수 있게 된다.
- 마찬가지로 컴퓨터에도 고유한 주소가 있다. 이를 IP 주소(Internet Protocol Address)라고 한다.

- IP 주소(IP Address)는 TCP/IP로 연결된 네트워크에서 각각의 컴퓨터를 구분하기 위해 사용하는 주소이다.
- IP 주소는 4개로 구분된 10진수 숫자로 구성되어 있고, 사용할 수 있는 IP 주소 범위는 아래와 같다.

구분명	법위	네트워크 수	사용 목적 / 네트워크 당 주소
클래스A	1.0.0.0 ~ 127.0.0.0	128	대형 통신망 / 17,777,214개
클래스B	128.0.0.0~191.255.0.0	16.384	중형 통신망 / 65,535개
클래스C	192.0.0.0~223.255.255.0	2,097,152	소형 통신망 / 256개
클래스D	224.0.0.0~239.255.255.255	-	멀티캐스트용
클래스E	240.0.0.0~255.255.255.255	-	실험 목적

IP 주소

- IP 주소는 네트워크 어댑터(LAN 카드)마다 할당된다.
- 만약 컴퓨터에 두 개의 네트워크 어댑터가 장착되어 있다면, 두 개의 IP 주소를 할당 받을 수 있다.
- 네트워크 어댑터에 어떤 IP 주소가 부여되어 있는지 확인하려면 터미널에 명령어를 입력하면 된다.
  - 윈도우는 ipconfig, 맥OS는 ifconfig 명령어를 입력

- 이론적으로는 컴퓨터가 IP 주소를 가지고 있어야 하나, 수많은 컴퓨터에 가기 다른 고유한 IP주소를 할당하기에는 한계가 있다.
- 모바일 인터넷의 급속한 확산과 함께 IP 주소 고갈의 문제는 네트워크와 라우터 기술의 발달로 사설 IP를 통해 완화되었다.

- 사설 IP는 일반적으로 10.0.0.0, 192.0.0.0 등의 몇 가지 IP 블록을 사용하지만, 실제로는 모든 인터넷 IP를 사용할 수 있다. 네트워크 구성 상 직접 연결하지는 않지만, 라우터 장비에서 제공하는 NAT(Network Access Tranlator) 기능을 이용한다. NAT를 이용해 공인 IP로 대응하면 인터넷에 접속할 수 있게 되는 것이다.

IP 주소

- 우리가 전화번호를 모르면 전화를 걸 수 없듯, 연결하고 싶은 컴퓨터의 IP 주소를 모르면 프로그램은 서로 통신할 수 없게 된다.
- 114에 전화번호를 문의하듯, 프로그램에서는 DNS(Domain Network System)를 이용해 컴퓨터의 IP 주소를 검색한다.

- 여기에서 DNS는 도메인 이름으로, IP를 등록하는 저장소이다.
- 대중에게 서비스를 제공하는 대부분의 컴퓨터는 도메인 이름으로 IP를 DNS에 미리 등록해놓는다.

- 웹 브라우저는 웹 서버와 통신하는 클라이언트로,
  사용자가 입력한 도메인 이름으로 DNS에서 IP 주소를 검색해 찾아 웹 서버와 연결하고 웹 페이지를 받아온다.

인터넷 기반 서비스

- 인터넷 기반의 대표적인 서비스는 웹 이외에도 이메일, FTP, Telnet, DNS 등이 있다.
- 인터넷 기반 서비스는 각각에 해당하는 프로토콜과 포트가 정해져 있다.

서비스명	기능	프로토콜	포트
웹(WWW)	웹 서비스	HTTP / HTTPS	80 / 443
이메일	이메일 서비스	SMTP / POP3 / IMAP	25 / 110 / 143
FTP	파일 전송 서비스	FTP	21
Telnet/SSH	원격 로그인 서비스	TELNET / SSH	23 / 22
DNS	도메인 이름 변환 서비스	DNS	53

Port

- 포트(Port)는 네트워크 서비스를 제공하기 위한 일종의 출입문이다.
- 즉, 하나의 컴퓨터에서 여러 개의 네트워크 서비스를 제공하는 경우, 이들을 구분하기 위한 목적으로 사용한다.

- 한 대의 컴퓨터에는 하나 이상의 다양한 서버 프로그램들이 실행될 수 있다.
- 웹 서버, 데이트베이스 관리 시스템, FTP 서버 등이 하나의 IP 주소를 갖는 컴퓨터에서 동시에 실행될 수 있다.

- IP는 컴퓨터의 네트워크 어댑터까지 갈 수 있는 정보이고,
  컴퓨터 내부에서는 실행하는 서버를 선택하기 위해 추가적인 Port 번호가 필요하다.
- 즉, Port는 운영체제가 관리하는 서버 프로그램의 연결 번호이다.

구분명	법위	설명
Well Know Port Numbers	0 ~ 1023	국제인터넷주소관리기구가 특정 애플리케이션용으로 미리 예약한 Port
Registered Port Numbers	1024 ~ 49151	회사에서 등록해서 사용할 수 있는 Port
Dynamic Or Private Port Numbers	49152 ~ 65535	운영체제가 부여하는 동적 Port 또는 개인적 목적으로 사용할 수 있는 Port

Port

- 서버는 시작할 때 특정 Port 번호에 바인딩된다. (예를 들어, 웹 서버는 80번으로, DBMS는 3306번으로 바인딩할 수 있다.)
- 프로그램에서 사용할 수 있는 전체 Port 번호의 범위는 0 ~ 65335로, 사용 목적에 따라 세 가지 범위를 가진다.

구분명	법위	설명
Well Know Port Numbers	0 ~ 1023	국제인터넷주소관리기구가 특정 애플리케이션용으로 미리 예약한 Port
Registered Port Numbers	1024 ~ 49151	회사에서 등록해서 사용할 수 있는 Port
Dynamic Or Private Port Numbers	49152 ~ 65535	운영체제가 부여하는 동적 Port 또는 개인적 목적으로 사용할 수 있는 Port

Port

- Java에서 IP 주소를 java.net 패키지의 InetAddress로 표현한다.
- InetAddress를 이용하면 로컬 컴퓨터의 IP 주소를 얻을 수 있고,
  도메인 이름으로 DNS에서 검색한 후 IP 주소를 가져올 수도 있다.

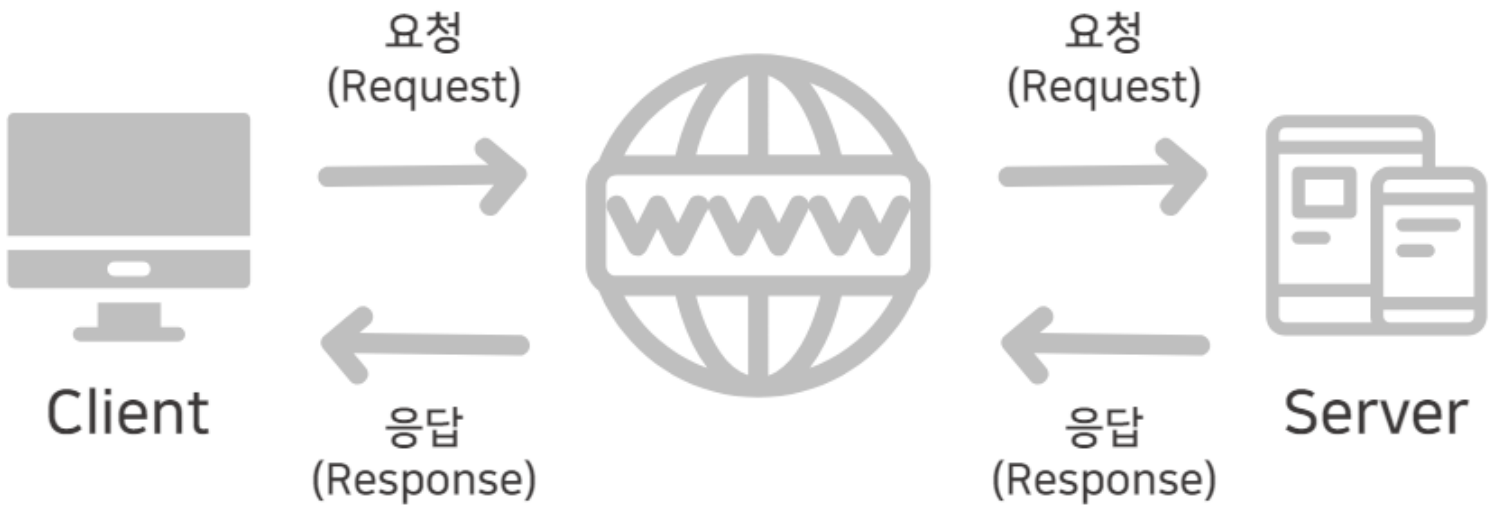
- 로컬 컴퓨터의 InetAddress를 얻고 싶다면 InetAddress.getLocalHost( ) 메소드를 다음과 같이 호출하면 된다.

```
public class IpExample {
    public static void main(String[] args) throws UnknownHostException {
        InetAddress local = InetAddress.getLocalHost();
        System.out.println("내 아이피: " + local.getHostAddress());

        InetAddress[] remoteArr = InetAddress.getAllByName("www.naver.com");
        for (InetAddress remote : remoteArr) {
            System.out.println("naver 아이피:" + remote.getHostAddress());
        }
    }
}
```

서버와 클라이언트

- 네트워크 안에 유무선으로 컴퓨터가 연결되어 있다면, 실제로 데이터를 주고받는 행위는 프로그램들이 한다.
- 서비스를 제공하는 프로그램을 일반적으로 서버(Server)라고 부르고,
  서비스를 요청하는 프로그램을 클라이언트(Client)라고 부른다.



JSON 데이터 형식

- 네트워크로 전달하는 데이터가 복잡할수록 구조화된 형식이 필요하다.
- 네트워크 통신에서 최근 가장 많이 사용되는 형식은 JSON(JavaScript Object Notation)이다.

- Java에서JSON을 문자열로 직접 작성할 수 있지만, 대부분은 라이브러리를 이용해서 생성한다.
- JSON을 다루는 라이브러리의 종류는 다양하고, 이에 따라 프로젝트마다 사용되는 라이브러리가 제각각이다.

https://www.json.org/json-ko.html

Web Application

- 웹 애플리케이션은 웹에서 실행되는 응용프로그램을 뜻하며, 인터넷을 통한 여러가지의 서비스를 일컫는다.

- 사용자는 필요한 데이터를 Request(요청)하고,
  서버에서는 사용자의 Request를 수행하며, 요청한 데이터를 Response(응답)하게 되는 형식이다.

- 웹 애플리케이션의 구성 요소

  1. 웹 브라우저 : 클라이언트에서 요청을 하고, 전달받은 페이지를 볼 수 있는 환경

  2. 웹 서버 : 서버에 저장된 리소스를 클라이언트에 전달하는 역할 (주로 정적 콘텐츠)

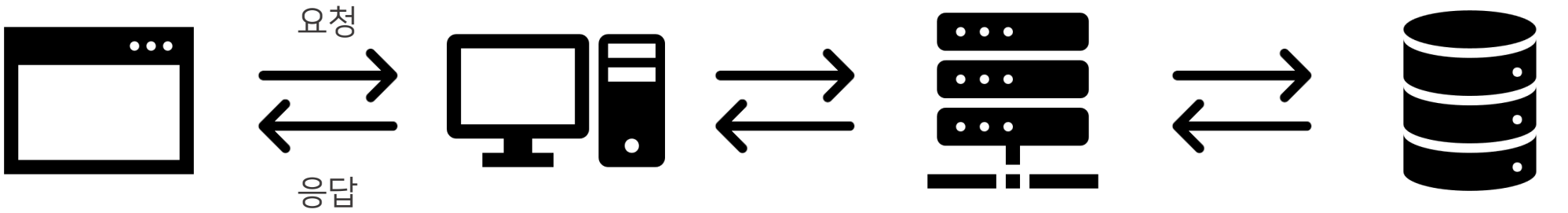
  3. 웹 애플리케이션 서버 : WAS라 부르며, 서버에서 필요한 기능을 수행하고, 그 결과를 웹 서버에 전달하는 역할

  4. 데이터베이스 : 서비스에 필요한 데이터를 보관, 갱신 등 관리를 수행하는 역할

Web Application

- 웹 애플리케이션의 구성 요소

1. 웹 브라우저 : 클라이언트에서 요청을 하고, 전달받은 페이지를 볼 수 있는 환경
2. 웹 서버 : 서버에 저장된 리소스를 클라이언트에 전달하는 역할 (주로 정적 콘텐츠)
3. 웹 애플리케이션 서버 : WAS라 부르며, 서버에서 필요한 기능을 수행하고, 그 결과를 웹 서버에 전달하는 역할
4. 데이터베이스 : 서비스에 필요한 데이터를 보관, 갱신 등 관리를 수행하는 역할



요청과 응답

- 서버는 클라이언트가 있기에 동작한다.

- 클라이언트에서 서버로 요청(request)을 보내고,
  서버에서는 요청의 내용을 읽고 처리한 뒤 클라이언트에 응답(response)을 보낸다.

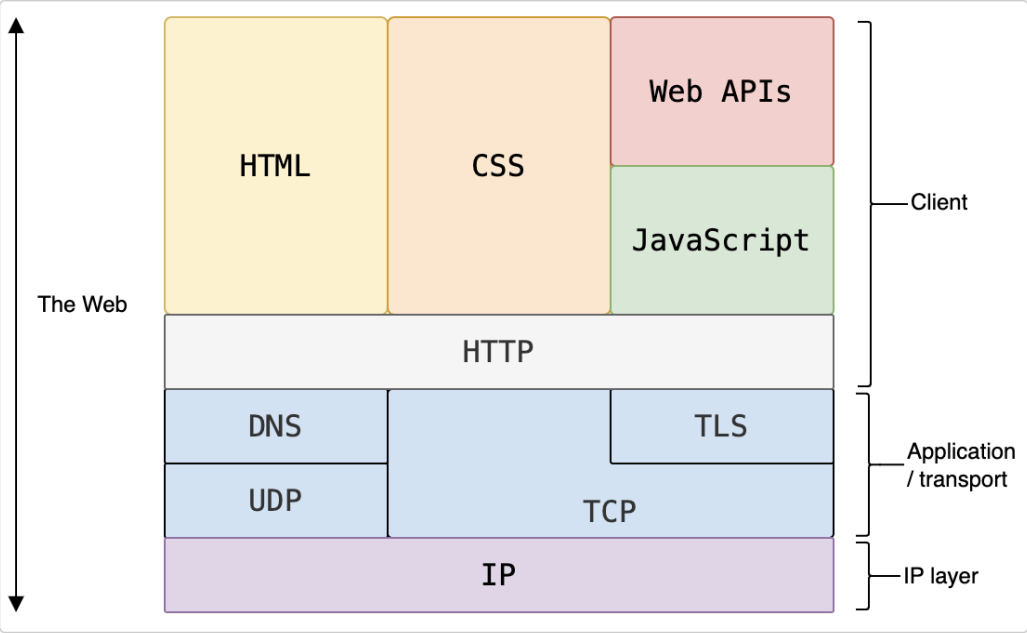
- 즉, 서버에는 요청을 받는 부분과 응답을 보내는 부분이 있어야 한다.

- 모든 요청과 응답은 헤더(Header)와 본문(Body)을 가지고 있다.
  - 헤더는 요청 또는 응답에 대한 정보를 가지고 있는 곳이고
  - 본문은 서버와 클라이언트 간에 주고받을 실제 데이터를 담아두는 공간이다.

HTTP

- HTTP는 HTML 문서와 같은 리소스들을 가져올 수 있도록 해주는 프로토콜이다.
- HTTP는 웹에서 이루어지는 모든 데이터 교환의 기초이며, 클라이언트-서버 프로토콜이기도 하다.

- 클라이언트와 서버들은 (데이터 스트림과 대조적으로) 개별적인 메시지 교환에 의해 통신한다.
- 클라이언트에 의해 전송되는 메시지를 요청(requests)이라고 부르며,
  그에 대해 서버에서 응답으로 전송되는 메시지를 응답(responses)이라고 부른다.



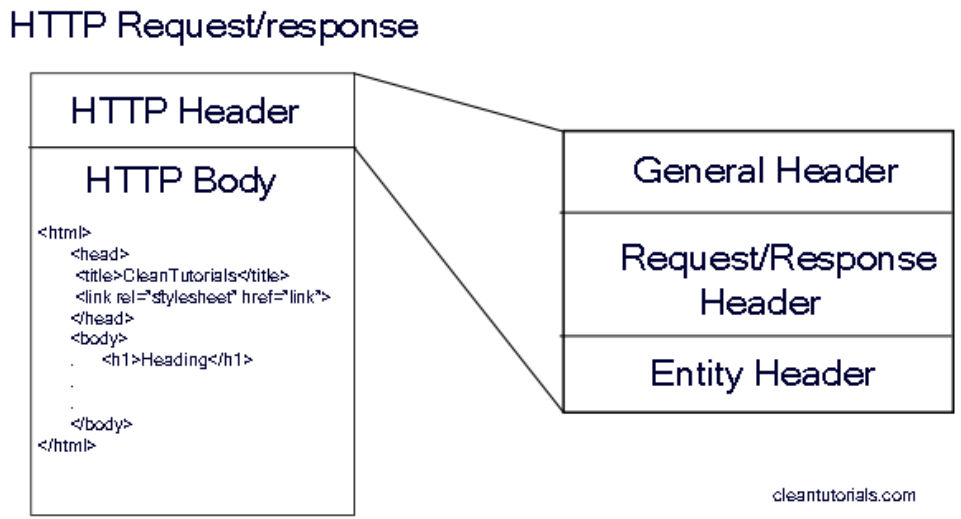
TIP

클라이언트-서버 프로토콜 :
수신자 측(주로 웹 브라우저)에 의해 요청이 초기화되는 프로토콜

HTTP 헤더

- HTTP 헤더는 클라이언트와 서버가 요청 또는 응답으로 부가적인 정보를 전송할 수 있도록 해준다.
- HTTP 헤더는 콜론(:)을 기준으로 키와 값으로 이루어져 있다.

  - 일반 헤더(General header) : 요청과 응답 모두에 적용되지만 Body에서 전송되는 데이터와는 관련이 없는 헤더.
  - 요청 헤더(Request header): Fetch될 리소스나 클라이언트 자체에 대한 자세한 정보를 포함하는 헤더.
  - 응답 헤더(Response header): 응답에 대한 부가적인 정보(위치 또는 서버 자체에 대한 정보 등)를 갖는 헤더.
  - 엔티티 헤더(Entity header): 콘텐츠 길이나 MIME 타입과 같이 Entity Body에 대한 자세한 정보를 포함하는 헤더.



TIP

HTTP 헤더의 키는 대소문자 구분을 하지 않으며,
값은 줄바꿈을 인식하지 않고,
맨 앞의 빈 문자열은 무시된다는 특징이 있다.

HTTP 상태 코드

- 웹 브라우저는 서버에서 보내주는 상태 코드를 보고 요청이 성공했는지 실패했는지를 판단한다.

- 200번대 코드 : 성공을 알리는 상태코드. [200(성공), 201(작성됨)]
- 300번대 코드 : 리다이렉션(다른 페이지로 이동)을 알리는 상태 코드. 다른 주소의 페이지로 넘어갈 때 이 코드가 사용된다.
- 400번대 코드 : 요청 자체에 오류가 있을 때 표시되는 상태 코드. [401(권한없음), 403(금지됨), 404(찾을 수 없음)]
- 500번대 코드 : 요청은 제대로 왔지만 서버에 오류가 생겼을 때 발생하는 상태 코드.

https://http.cat/상태코드

HTTP 요청 메소드

- 클라이언트가 서버에 데이터를 전송하여 응답을 요청할 때 사용하는 방식

- GET : 서버 자원을 가져오고자 할 때 사용하며, 요청의 본문에 데이터를 넣지 않는다.
  데이터를 서버로 보내야 한다면 쿼리스트링을 사용한다. (?name=최인규&age=20)

- POST : 서버에 자원을 새로 등록하고자 할 때 사용하며, 요청의 본문에 새로 등록할 데이터를 넣어 보낸다.

- PUT : 서버의 자원을 요청에 들어 있는 자원으로 치환하고자 할 때 사용하며, 요청의 본문에 치환할 데이터를 넣어 보냅니다.

- PATCH : 서버 자원의 일부만 수정하고자 할 때 사용하며, 요청의 본문에 일부 수정할 데이터를 넣어 보냅니다.

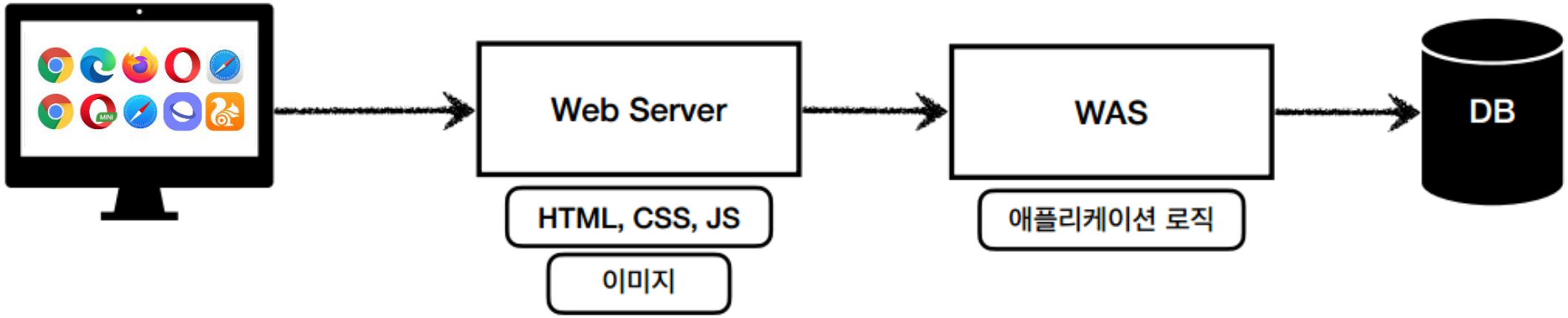
- DELETE : 서버의 자원을 삭제하고자 할 때 사용하며. 요청의 본문에 데이터를 넣지 않는다.

- OPTIONS : 요청을 하기 전에 통신 옵션을 설명하기 위해 사용한다.

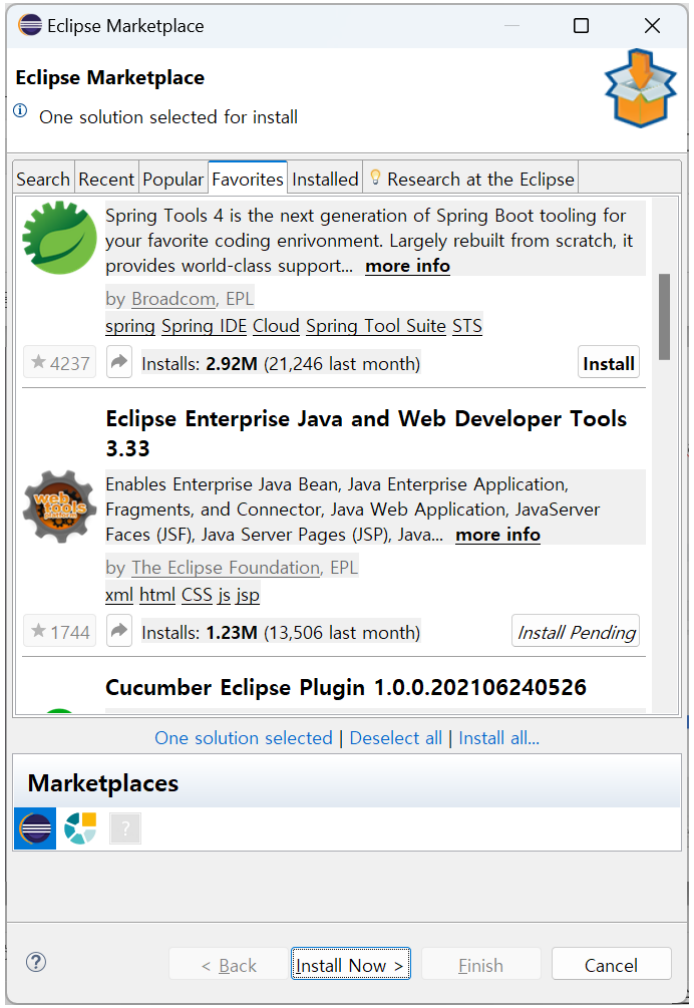
서블릿 컨테이너 (Servlet Container)

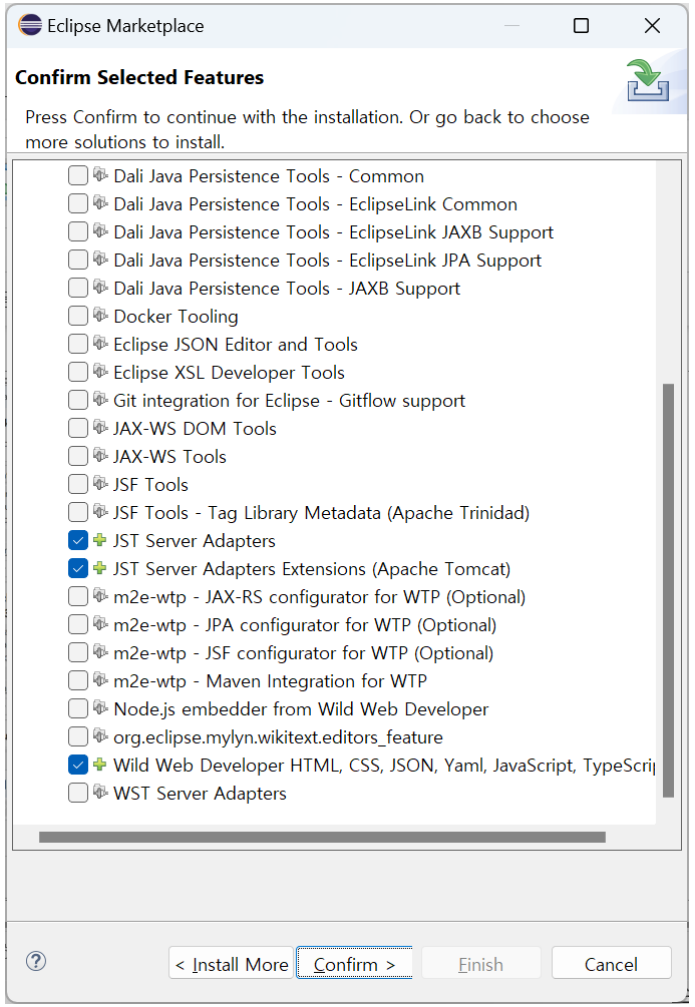
- WAS(Web Application Server)는 웹 애플리케이션을 구동하는 서버를 의미한다.
- 서버 컴퓨터가 WAS로 동작하기 위해서는 Java EE 또는 아파치 톰캣(Tomcat)과 같은 서블릿 컨테이너가 필요하다.

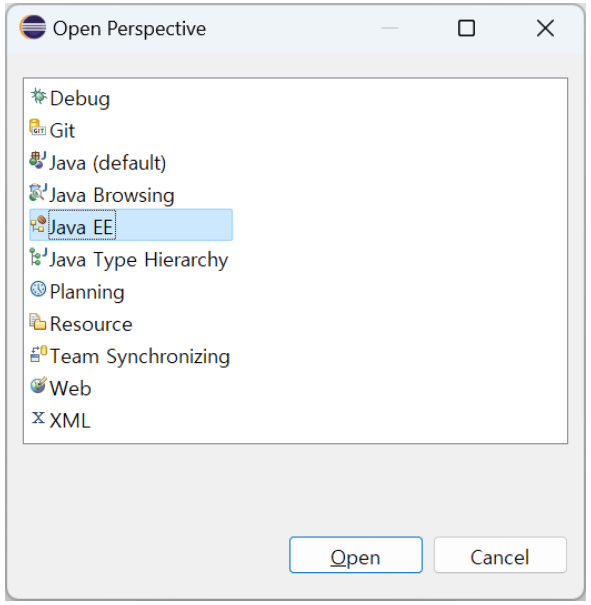
- WAS는 단순히 Java EE 또는 서블릿 컨테이너의 구동을 위해 존재하는 것은 아니고,
  운영 및 관리, 장애 대응 등 여러 가지 역할을 하는 시스템이다.

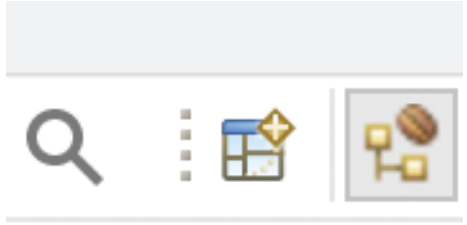


웹 애플리케이션 개발 환경 구축 (Eclipse EE 설치)









웹 애플리케이션 개발 환경 구축 (Eclipse EE 설치)

- 텍스트 인코딩 설정 확인
  - Window - Preferences
    - General - Content Type - Text
      - Default encoding : UTF-8

- HTML, CSS, JSP의 인코딩 설정 확인
  - Window - Preferences
    - Web - HTML Files
      - Encoding : ISO 10646/Unicode(UTF-8)
    - Web - CSS Files
      - Encoding : ISO 10646/Unicode(UTF-8)
    - Web - JSP Files
      - Encoding : ISO 10646/Unicode(UTF-8)

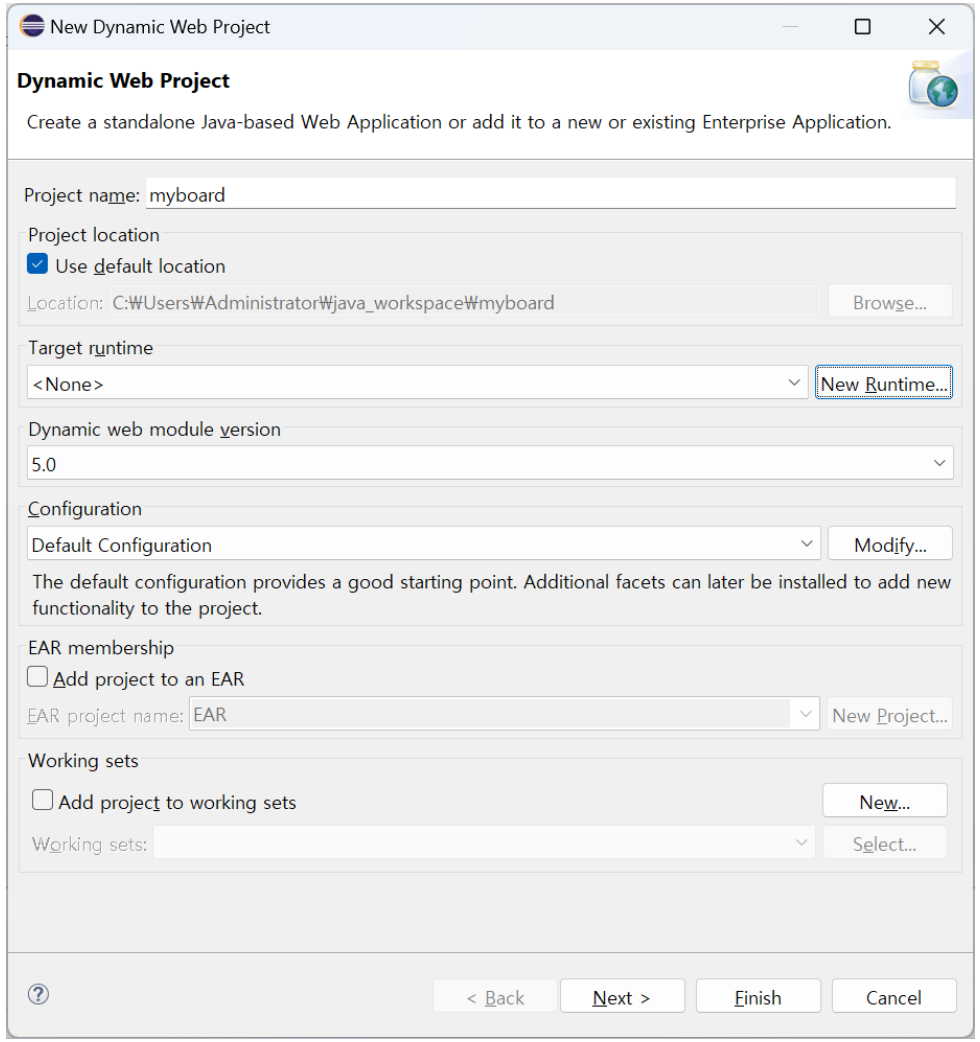
- 웹 브라우저 설정
  - Window - Preferences
    - General - Web Browser
      - Chrome 선택

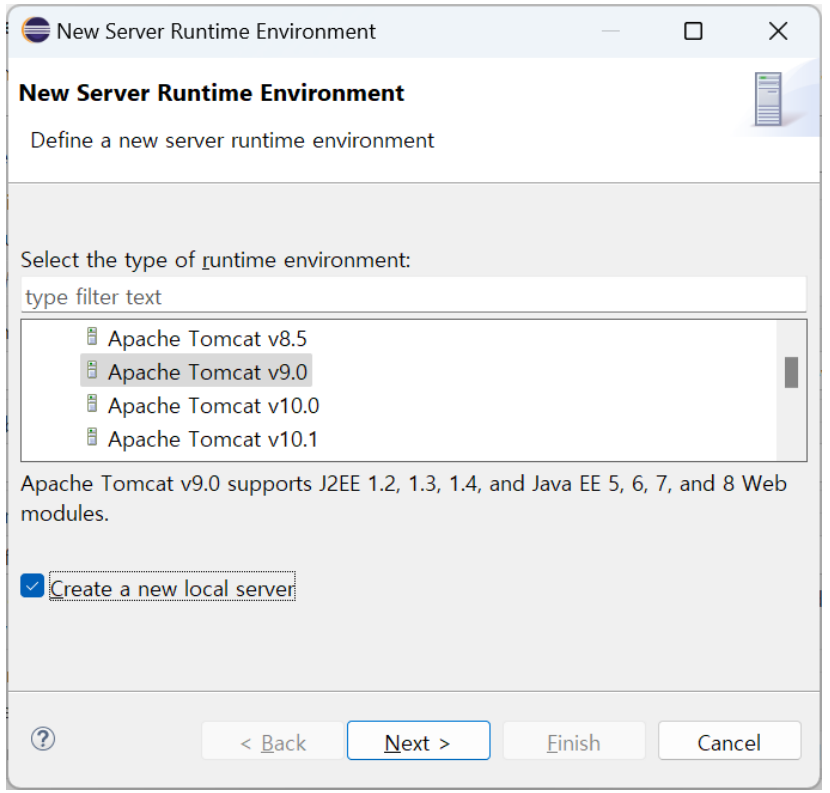
웹 애플리케이션 개발 환경 구축 (TOMCAT 9 설치)

- Java EE가 이클립스 재단으로 이관되면서, 그 명칭이 변경되었고, 패키지 체계가 javax에서 jakarta로 변경되었다.

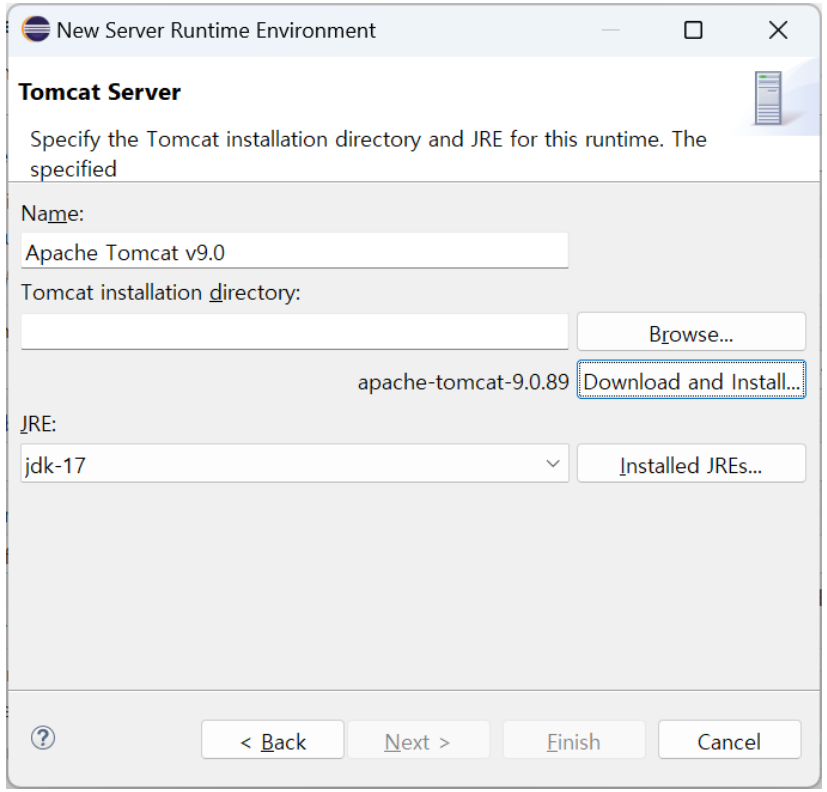
- TOMCAT 9 버전에서는 javax.* 패키지를 따르고 있으며, 10 버전부터는 jakarta.* 패키지를 따르고 있다.

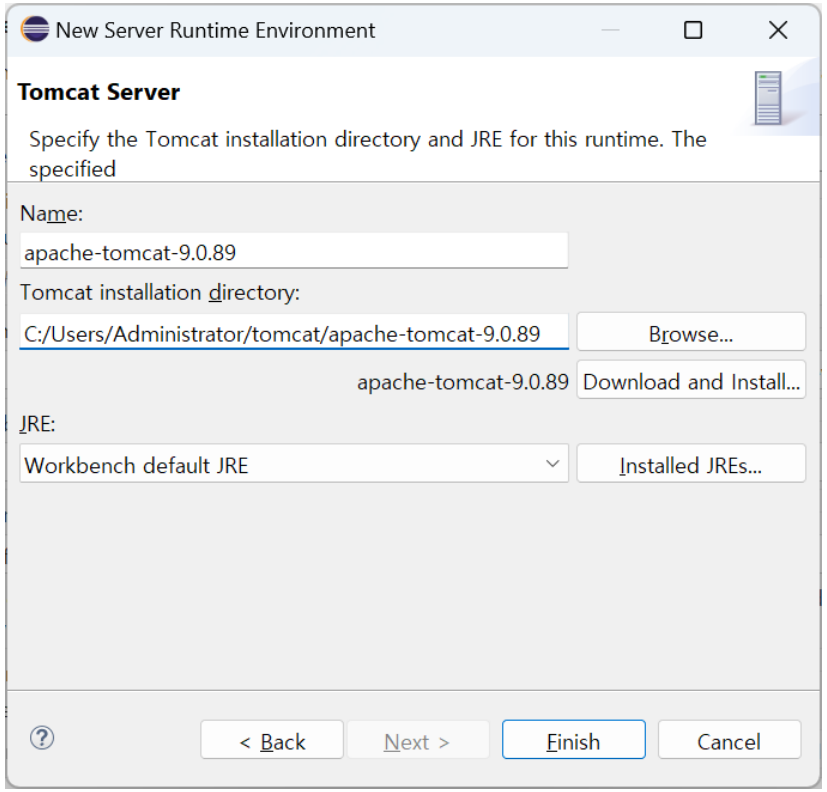
웹 애플리케이션 개발 환경 구축 (TOMCAT 9 설치)



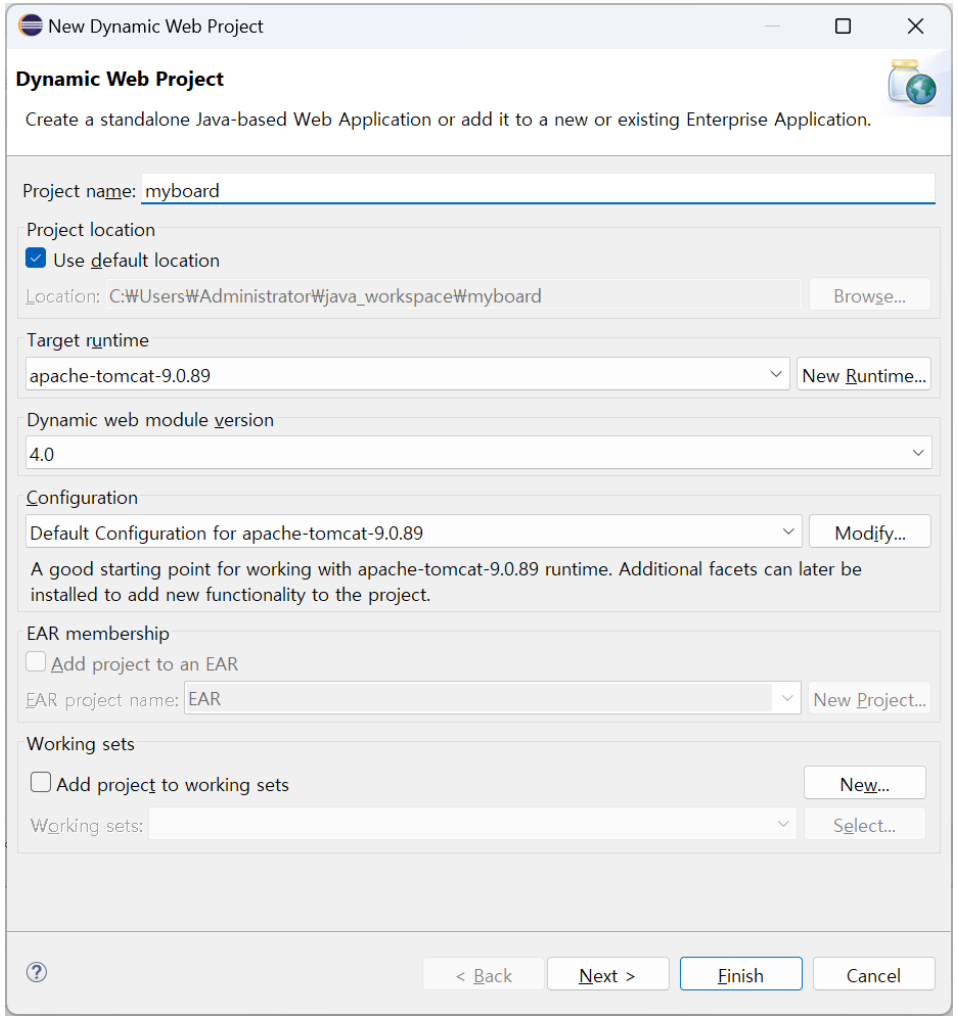


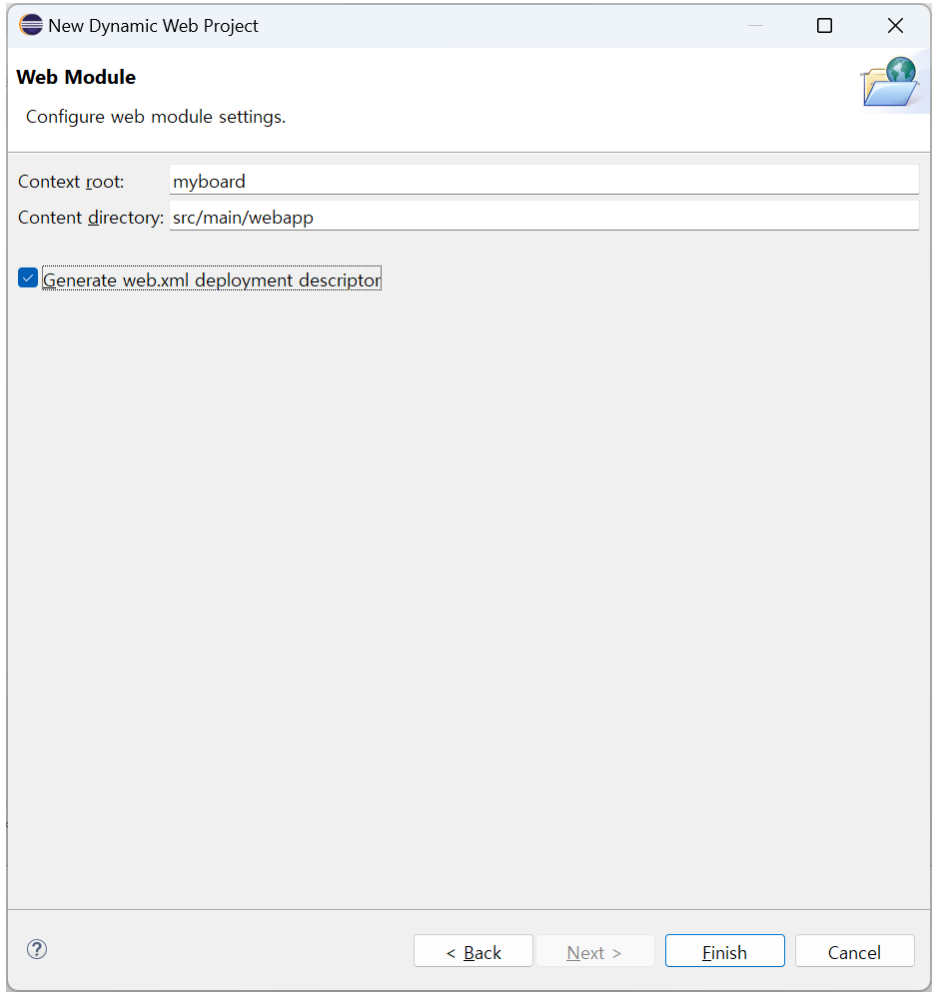
웹 애플리케이션 개발 환경 구축 (TOMCAT 9 설치)



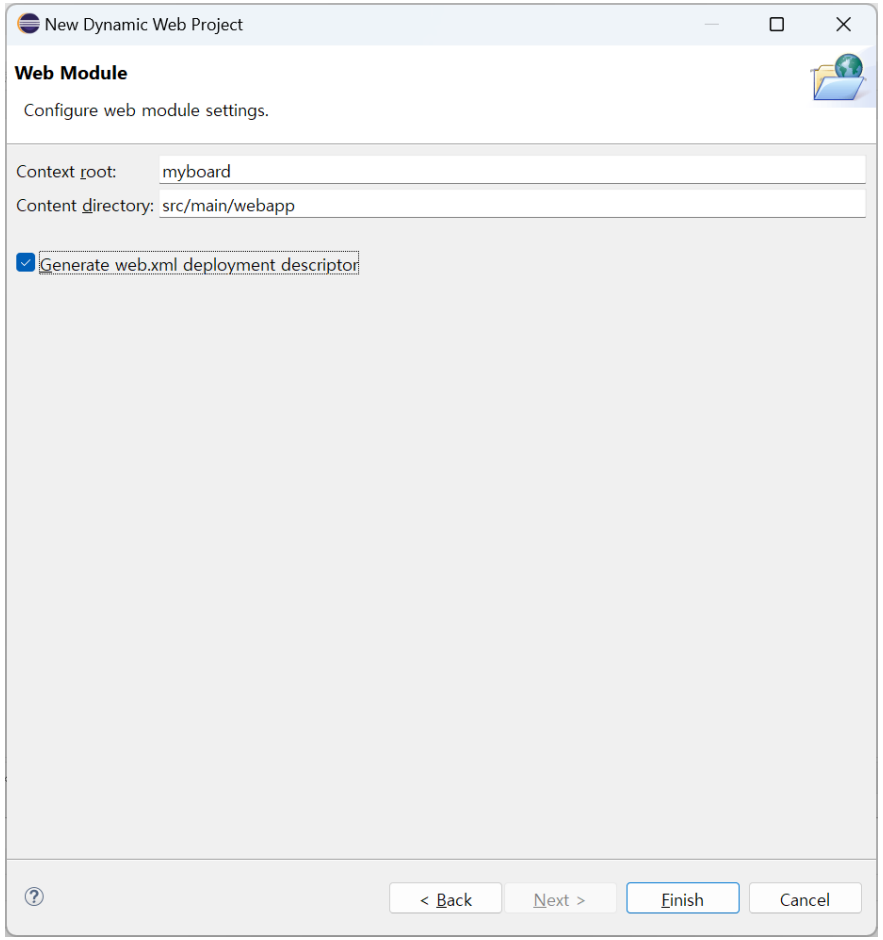


웹 애플리케이션 개발 환경 구축 (TOMCAT 9 설치)



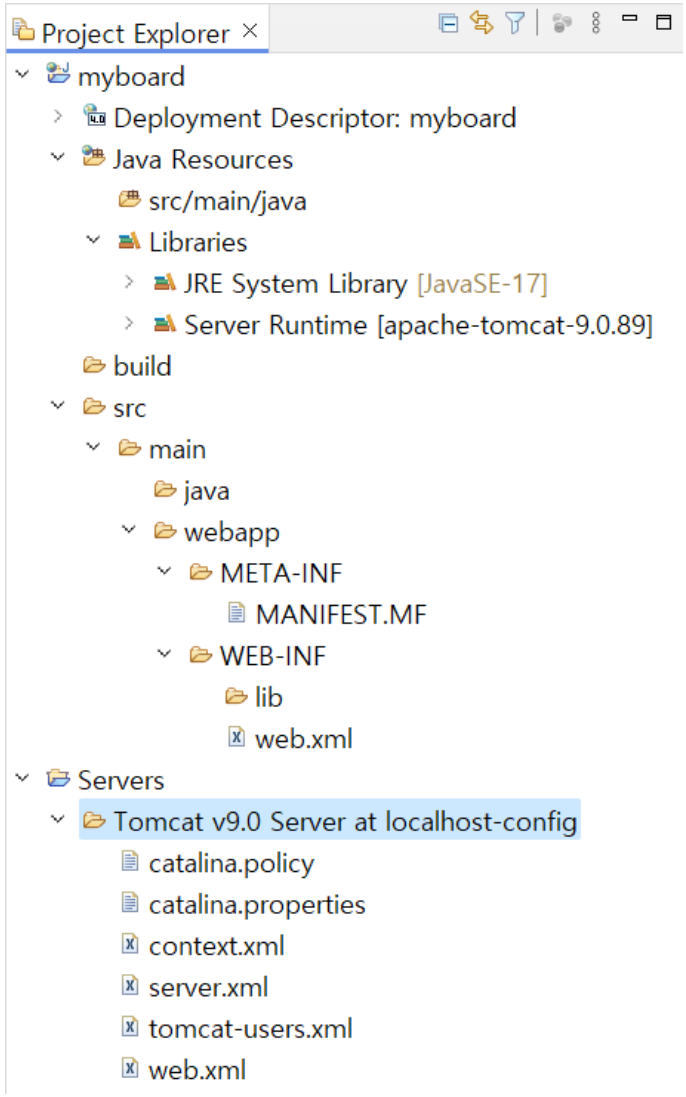


웹 애플리케이션 개발 환경 구축 (TOMCAT 9 설치)



- Context root : 톰캣을 통해 실행할 때 사용되는 진입점
- Content directory : 웹 콘텐츠(HTML, CSS, JS)가 위치하는 경로
- web.xml : 웹 애플리케이션 관련 정보를
  서블릿 컨테이너에 제공하기 위한 설정 파일 (옵션 사항)

웹 프로젝트 시작하기



- src/main/webapp
  - HTML, JSP, Javascript, CSS 및 이미지 파일들이 위치한다.
    해당 위치에 놓이는 파일들은 웹 애플리케이션이 배치될 때 그대로 옮겨진다.

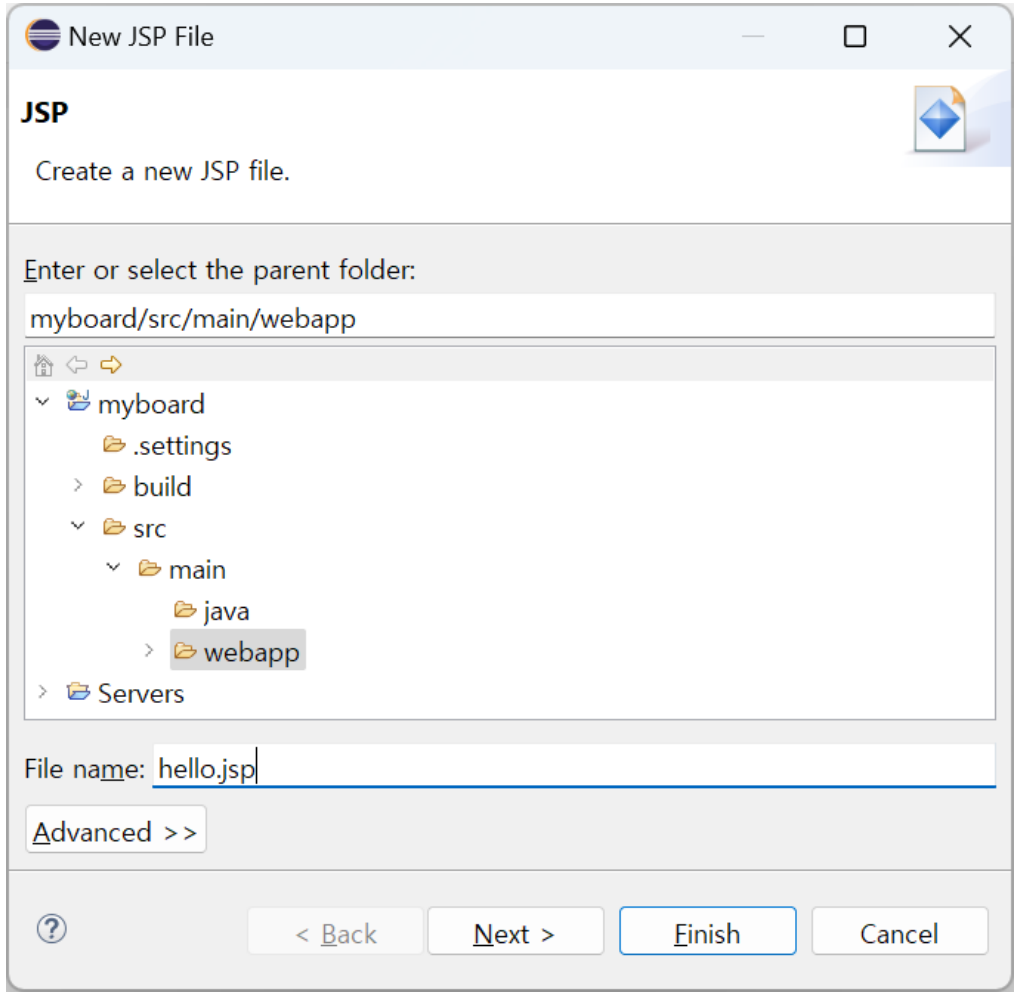
- src/main/webapp/WEB-INF
  - 웹 애플리케이션의 설정 파일들이 이곳에 포함된다.
    WEB-INF 내에 있는 파일들은 클라이언트에서 요청할 수 없다.
    따라서 HTML, JS와 같은 정적 자원은 바로 읽을 수 없다.

- src/main/webapp/WEB-INF/classes
  - 컴파일된 자바 클래스(class) 파일이 있으며 src에 선언된 package가 동일하게 생성된다.

- src/main/webapp/WEB-INF/lib
  - 실행에 필요한 라이브러리(jar 파일)들을 모아두는 디렉토리이다.

- src/main/webapp/WEB-INF/web.xml
  - 웹 애플리케이션 배치 설명서 파일로
    서블릿 컨테이너는 web.xml를 참조해서 서블릿, 필터, 매개변수 등을 찾고 실행한다.

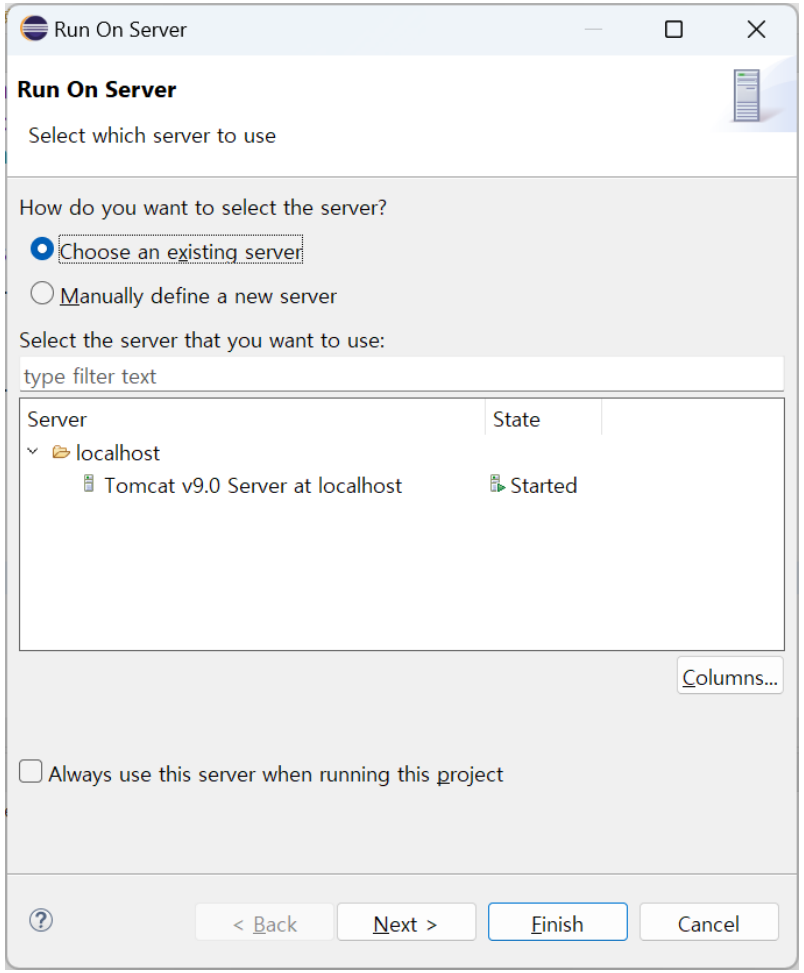
웹 프로젝트 시작하기

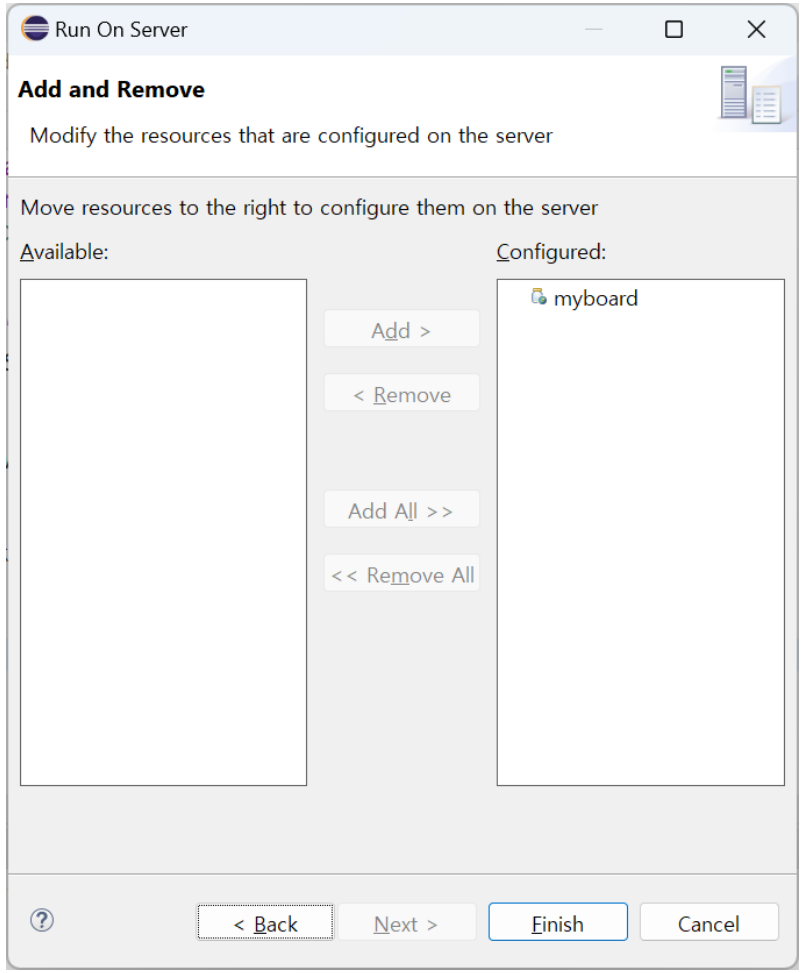

New JSP File
JSP
Create a new JSP file.
Enter or select the parent folder:
myboard/src/main/webapp
myboard
.settings
build
src
main
java
webapp
Servers
File name: hello.jsp
Advanced >>
< Back
Next >
Finish
Cancel

웹 프로젝트 시작하기

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Hello JSP</title>
</head>
<body>
    <h1>Hello World~!</h1>
    <hr>
    <p>
        현재 날짜와 시간은 <%=java.time.LocalDateTime.now() %> 입니다.
    </p>
</body>
</html>
```

웹 프로젝트 시작하기


Run On Server
Run On Server
Select which server to use
How do you want to select the server?
Choose an existing server
Manually define a new server
Select the server that you want to use:
type filter text
Server
State
localhost
Tomcat v9.0 Server at localhost
Started
Columns...
Always use this server when running this project
< Back
Next >
Finish
Cancel


Run On Server
Add and Remove
Modify the resources that are configured on the server
Move resources to the right to configure them on the server
Available:
Configured:
myboard
Add >
< Remove
Add All >>
<< Remove All
< Back
Next >
Finish
Cancel

서블릿 (Servlet)

- 서블릿이란 Java 기반의 Web 프로그램 개발을 위해 만들어진 기술이다.
- 서블릿을 실행하기 위해서는 톰캣과 같은 서블릿 컨테이너가 필요하며, 이를 일반적으로 WAS라 부른다.

- 데스크톱이나 스마트폰에서 실행되는 일반적인 애플리케이션과 달리 웹 프로그램은 서버에서 접속해야 화면을 볼 수 있다.

- 이 때 HTML은 정적 파일이므로 실시간으로 변하는 뉴스 기사, 날씨 정보, 쇼핑몰의 상품 정보 등을 가지고 있을 수 없다.
- 이러한 정보는 DB에 저장되어 있기 때문에 사용자의 웹 요청을 받으면
  별도의 프로그래밍을 통해 DB에서 가져와 HTML로 재구성하여 클라이언트에 다시 전달해야 한다.

서블릿 (Servlet)

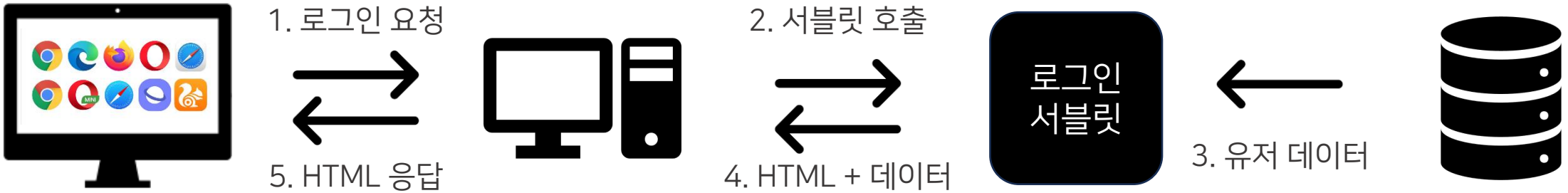
- 웹 요청과 응답 과정은 다음과 같다.

    - 클라이언트(웹 브라우저)가 서버에 페이지(index.html)를 요청한다.
    - 서버는 클라이언트에 요청받은 파일(html)을 응답한다.
    - 클라이언트는 수신받은 파일(html) 내용을 해석(CSS, JavaScript)하여, 화면에 표시한다.

- 이 때, HTML을 정적 파일이므로, 실시간으로 변하는 정보 등을 가지고 있을 수 없다.
- 변경된 정보를 표시하고 싶다면, 다시 요청을 보내 DB에서 가지고 온 정보로 재구성된 HTML로 클라이언트에 재전달해야 한다.

서블릿 (Servlet)

- 이 외에 사용자 로그인 등과 같은 처리를 위해서도 마찬가지로,
  입력한 아이디, 비밀번호를 데이터베이스에 저장된 내용과 비교하고 결과를 전달받아야 한다.



- 서블릿은 이러한 기능들을 수행할 수 있도록 설계된 특수한 목적의 자바 프로그램이다.

서블릿 (Servlet)

- Java를 기반으로 하는 서블릿의 장점은 아래와 같다.

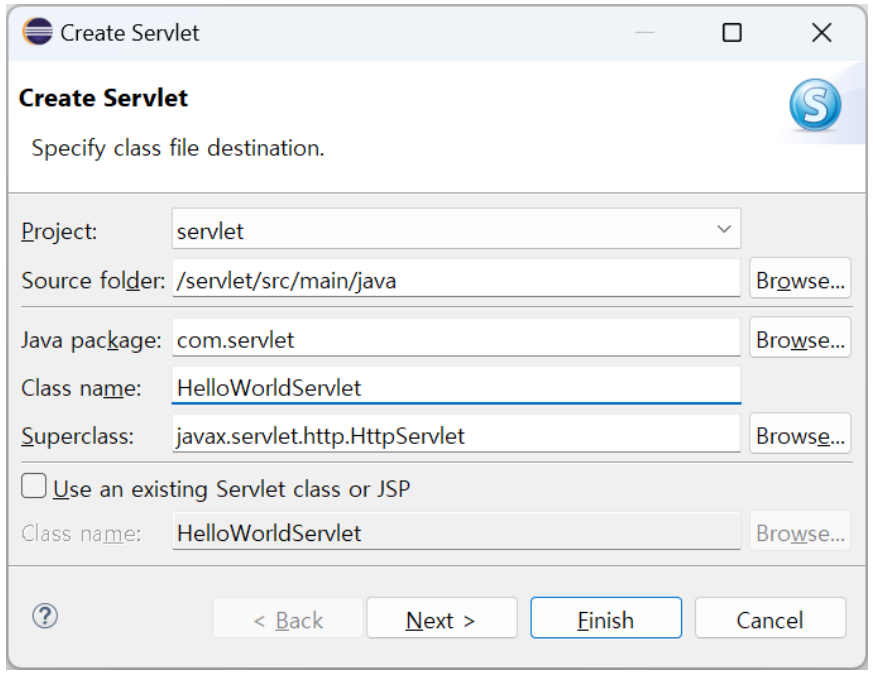
  - Java API를 모두 사용할 수 있다.
  - 운영체제나 하드웨어의 영향을 받지 않으므로, 한 번 개발된 애플리케이션은 다양한 서버 환경에서도 실행 가능하다.
  - 다양한 오픈소스 라이브러리와 개발 도구를 활용할 수 있다.

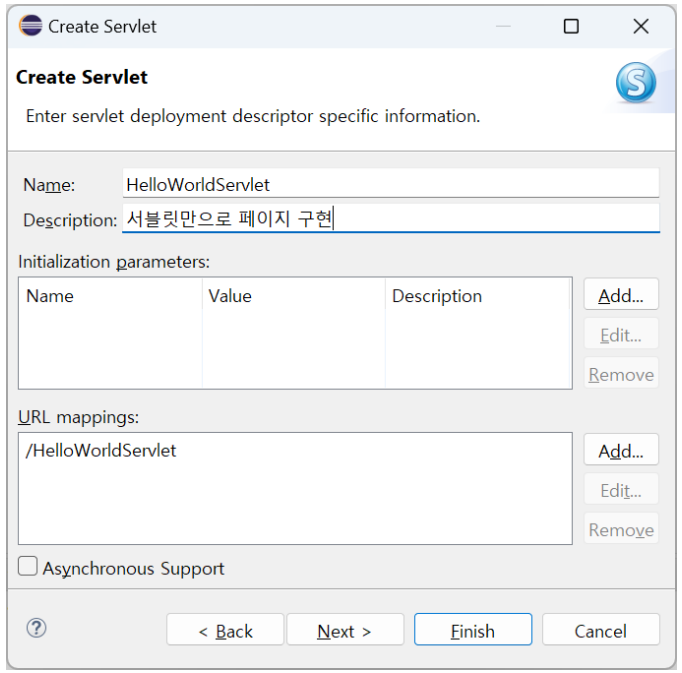
- 반면 서블릿의 단점은 아래와 같다.

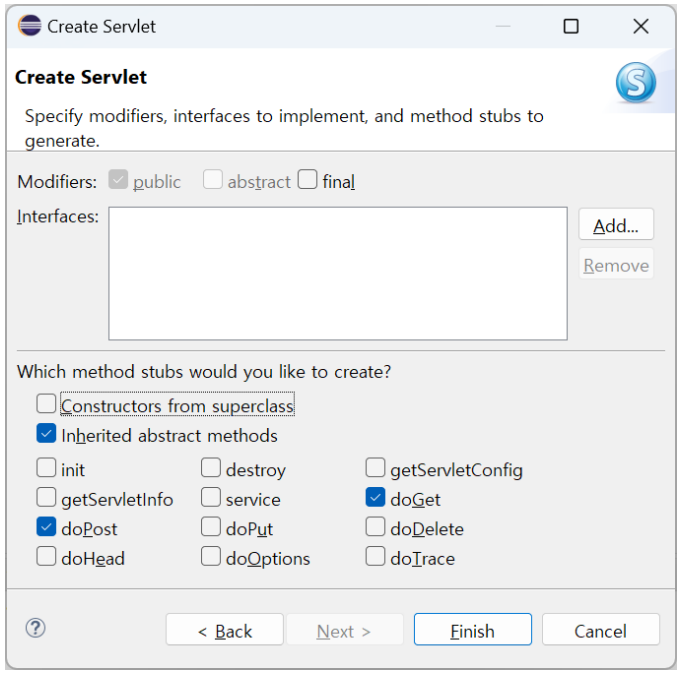
  - HTML 응답을 위해서는 출력문으로 문자열 결합을 사용해야 한다.
  - 서블릿에 HTML이 포함된 경우, 화면 수정이 어렵다.
  - HTML 폼 데이터 처리가 불편하다.
  - 기본적으로 단일 요청과 응답을 처리하는 구조이기 때문에 다양한 경로의 URL 접근을 하나의 클래스에서 처리하기 어렵다.

- 위와 같은 단점을 보완하기 위해 JSP가 생겨났으며,
  HTML 문서 내에 Java 코드를 삽입하는 방식이기 때문에 HTML 파일을 직접 수정하는 것이 가능해졌다.

서블릿 (Servlet)


Create Servlet
Create Servlet
Specify class file destination.
Project: servlet
Source folder: /servlet/src/main/java
Browse...
Java package: com.servlet
Browse...
Class name: HelloWorldServlet
Superclass: javax.servlet.http.HttpServlet
Browse...
Use an existing Servlet class or JSP
Class name: HelloWorldServlet
Browse...
< Back
Next >
Finish
Cancel


Create Servlet
Create Servlet
Enter servlet deployment descriptor specific information.
Name: HelloWorldServlet
Description: 서블릿만으로 페이지 구현
Initialization parameters:
Name
Value
Description
Add...
Edit...
Remove
URL mappings:
/HelloWorldServlet
Add...
Edit...
Remove
Asynchronous Support
< Back
Next >
Finish
Cancel


Create Servlet
Create Servlet
Specify modifiers, interfaces to implement, and method stubs to generate.
Modifiers: public abstract final
Interfaces:
Add...
Remove
Which method stubs would you like to create?
Constructors from superclass
Inherited abstract methods
init
destroy
getServletConfig
getServletInfo
service
doGet
doPost
doPut
doDelete
doHead
doOptions
doTrace
< Back
Next >
Finish
Cancel

서블릿 (Servlet)

```java
@WebServlet(description = "서블릿만으로 페이지 구현", urlPatterns = { "/HelloWorldServlet" })
public class HelloWorldServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("</html>");
    }

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

서블릿 (Servlet)

- 서블릿 클래스의 구조를 살펴보자.

- 서블릿 자체는 Java로 구현하지만 서블릿 컨테이너에 해당 클래스가 서블릿임을 알려야 하며,
  어떤 URL 접근에 실행해야 하는지 등록하는 과정이 필요하다.

- 과거에는 배포 서술자인 web.xml에 앱 애플리케이션 구조를 등록해야 했지만,
  서블릿 3.0부터는 별도의 web.xml 작성 없이 자바 애너테이션을 이용한 방법이 주로 사용되고 있다.

- 서블릿 클래스는 Serlvet 인터페이스를 구현한 추상 클래스인
  GenericServlet 클래스 또는 HttpServlet 클래스 중 하나를 상속해 구현된다.
- 웹 프로그래밍의 대부분은 HTTP 프로토콜에 최적화되어 있는 HttpServlet 클래스를 상속해 구현된다.

서블릿 (Servlet)

- HttpServletRequest는 HTTP 프로토콜의 요청 정보를 서블릿에 전달하기 위한 목적으로 사용된다.

- HttpServletRequest의 주요 메소드는 아래와 같다.

메소드	설명
getParameter(name), getParameterValues(name)	name 속성으로 전달된 값
getRequestURL(), getRequestURI()	URL, URI
getScheme(), getServerName(), getServerPort()	프로토콜, 서버 이름, 서버 포트
getContextPath()	콘텍스트 경로
getMethod()	HTTP 메소드
isSecure()	SSL 보안 여부
getLocale(), getLocalAddr(), getRemoteAddr()	지역 정보, 로컬 IP 주소, 클라이언트 IP 주소

서블릿 (Servlet)

- HttpServletResponse는 요청을 보낸 클라이언트로 응답하기 위한 목적으로 사용된다.

- HttpServletResponse의 주요 메소드는 아래와 같다.

메소드	설명
sendRedirect(location)	클라이언트에 리다이렉트 응답을 보낸 후, 특정 위치로 다시 요청하게 함
getWriter()	클라이언트로 데이터를 보내기 위한 출력 스트림 반환
setContentType(type)	클라이언트로 전달하는 콘텐츠 타입 지정
addCookie(cookie)	응답에 쿠키를 추가
addHeader(name, value)	헤더에 name과 value를 추가
encodeURL(url)	클라이언트가 쿠키를 지원하지 않을 때 세션 id를 포함한 특정 URL을 인코딩
getHeaderNames()	현재 응답이 헤더에 포함된 name을 얻어옴

서블릿 (Servlet)

- 서블릿은 컨테이너에 의해 동작하므로 객체의 생성 과정과 종료 과정도 컨테이너 안에서 이뤄진다.

- 서블릿 객체의 생성과 종료에 이르는 과정을 서블릿 생명주기라고 한다.
- 서블릿 클래스는 기본적으로 doGet(), doPost()와 같은 HTTP 요청 메소드에 따라 필요한 메소드를 오버라이딩해 구현하며, 컨테이너에 의해 객체의 생성과 소멸 등이 관리되므로 필요에 따라 특정 생명 주기 이벤트에 동작하는 메소드를 구현해야 한다.

메소드	설명
`init()`	서블릿을 메모리에 적재하는 과정에 실행되기 때문에 초기에 한 번만 실행된다. (서블릿 초기화 이벤트)
`service()` `doGet(), doPost()`	service() 메소드를 통해 각각 doGet() 이나 doPost()로 분기된다.
`destroy()`	컨테이너로부터 서블릿 종료 요청이 있을 때, destroy() 메소드를 호출한다. (서블릿 종료 시 정리해야 하는 작업)

서블릿 (Servlet)

- 사용자 요청을 처리한 후에는 적절한 화면으로 넘어갈 수 있어야 한다.

- 만약 별도의 데이터를 포함하지 않는다면, 해당 페이지로 바로 리다이렉션할 수 있다.
  세션에 데이터를 저장한 경우라면, 세션이 유효한 동안 리다이렉션을 통해서도 데이터 참조가 가능하다.

```
response.sendRedirect("main.jsp");
```

- 데이터를 포함하여 이동해야 하는 경우라면, request 속성으로 데이터를 넣은 후 원하는 페이지로 포워딩해야 한다.

```
request.setAttribute("data", "데이터");
RequestDispatcher dispatcher = request.getRequestDispatcher("index.jsp");
dispatcher.forward(request, response);
```

서블릿 (Servlet)

- 쿠키는 클라이언트에 저장되는 작은 정보를 의미한다.
- 서버의 요청에 의해 브라우저가 저장하게 되며, 서버가 요청할 때 제공하는 형식이다.

- 쿠키의 특징은 아래와 같다.
  - 파일로 클라이언트 컴퓨터에 저장되는 방식으로 보안상 문제가 될 수 있다.
  - 광고 혹은 기타 목적으로 사용자 이용 행태 추적에 이용될 수 있으며, 이러한 경우에는 사용자 동의가 필요하다.
  - 연속되는 페이지 이동에 대한 정보보다는 재방문 등의 확인 용도로 많이 사용된다.
  - 'name=value' 형식이며 유효 기간, 요청 경로, 도메인 지정 등의 부가 속성을 포함한다.
  - 주로 자바스크립트를 통해 처리하지만 HttpOnly 설정으로 서버에서만 사용할 수 있도록 설정할 수 있다.

서블릿 (Servlet)

```
response.addCookie(new Cookie("cookie", "쿠키"));
RequestDispatcher dispatcher = request.getRequestDispatcher("index.jsp");
dispatcher.forward(request, response);
```

```
<%
    Cookie[] cookies = request.getCookies();
    for(Cookie c: cookies) {
        out.print("쿠키 이름 " + c.getName() + "<br>");
        out.print("쿠키 값 " + c.getValue() + "<br>");
        out.print("-------------------------------<br>");
    }
%>
```

서블릿 (Servlet)

- 세션은 클라이언트가 웹 애플리케이션 서버에 접속할 대 서버 쪽에 생성되는 공간이다.
- 내부적으로는 세션 아이디를 통해 참조된다.

- 즉, 브라우저는 서버에 접속할 때 발급받은 세션ID를 기억하고 서버는 해당 세션ID로 할당된 영역에 접근하는 형식이다.

- 세션의 특징은 아래와 같다.
  - 세션 유효 시간이나 브라우저 종료 전까지 유지되므로 서로 다른 페이지에서도 정보 공유가 가능하다.
  - 로그인 유지, 장바구니, 컨트롤러 구현 등에서 다양하게 사용된다.
  - 사용자마다 생성되는 공간으로, 동시에 많은 사용자가 접속하는 경우 충분한 메모리를 비롯한 세션 관리 대책이 필요하다.

서블릿 (Servlet)

```
HttpSession session = request.getSession();
session.setAttribute("session", "세션");
RequestDispatcher dispatcher = request.getRequestDispatcher("index.jsp");
dispatcher.forward(request, response);
```

```
<%
    out.print("세션 값: " + (String) session.getAttribute("session") + "<br>");
%>
```

JSP

- JSP는 Java Server Page or Jarkarta Server Page를 뜻하며,
  HTML 코드에 Java 코드를 사용하여, 동적 웹 페이지를 생성하는 웹 어플리케이션 도구(라이브러리)이다.

- JSP는 서블릿에서 HTML과 데이터 결합을 손쉽게 처리하기 위해 만들어졌다.

- 이는 HTML에 Java 코드를 더한 형태(HTML이 Java 코드를 품고 있는 형태)로,
  컨테이너에 의해 서블릿 형태의 자바 코드로 변환 후 컴파일되어 컨테이너에 적재되는 구조이다.

- 이러한 편리성 덕분에 오랫동안 JSP는 웹 개발의 기본이 되어왔다.

JSP

- 그러나 JSP를 이용한 개발에도 몇 가지 문제가 발생되는데,
  게시판이나 상품 목록과 같이 데이터를 반복해서 출력하거나 조건을 체크해야 하는 경우에는 단순한 HTML 문법으로 처리가 불가능하기 때문에 Java 코드를 HTML 내부에 중간중간 사용해야 한다는 것이다.

JSTL/EL

- JSP의 구조적 문제를 해결하기 위해
  커스텀 태그를 기반으로 하는 JSTL(JSP Standard Tag Library)과 EL(Expression Language)가 도입되었다.

- JSP보다 훨씬 구조적이고 가독성도 높아졌으며,
  이와 같은 프로그램 구조는 MVC 패턴과 함께 서블릿과 결합되어 오랫동안 Java 웹 개발의 정석으로 자리 잡아왔다.

- 이러한 구조에도 단점은 있는데, SSR 구조(서버에서 빌드하는 구조)이기 때문에
  모든 실행이 서블릿 컨테이너를 통해야 한다는 점이다.
- 따라서 사소한 디자인 변경도 서버를 통해 실행해야 하며, 이는 개발 생산성과 디자이너와의 협업 등에서 큰 문제가 되게 된다.

커스텀 태그

- 커스텀 태그란 사용자 정의 태그를 의미한다.
- 즉, 스크립트릿(<% %>의 사용을 줄이고,
  태그와 같은 형태로 프로그램 코드를 대체하거나 재활용 가능한 구조를 활용하고자 개발된 규적이다.

- 커스텀 태그의 외형적인 형태는 XML(HTML) 태그 구조이지만, 서블릿 형태로 변환될 때 자바 코드로 변경되어 통합된다.

- 커스텀 태그를 사용하기 위해서는 taglib 지시어를 사용하여, 커스텀 태그가 어디에 정의되어 있는지 먼저 선언해야 하며,
  태그에 사용할 접두어 또한 지정해야 한다.

- 개발자가 직접 커스텀 태그를 만드는 방식은 점차 사용되지 않게 되고,
  커스텀 태그 기술로 만들어진 JSTL(JSP Standard Tag Library)이 사용되게 되었다.

EL

- EL(Expression Language)는 현재 페이지의 Java 객체에 손쉽게 접근하고 사용할 수 있게 해준다.

- 기본적으로 데이터를 표현하기 위한 용도로 설계되었지만
  EL은 단순한 출력 외에도 제한된 객체 참조가 가능하며, 해당 객체의 메서드 호출, 그리고 사칙, 비교, 3항 연산 등을 지원한다.

- EL의 장점은 아래와 같다.
  - 간단한 구문으로 손쉽게 변수/객체를 참조할 수 있다.
  - 데이터가 없거나 null 객체를 참조할 때 에러가 발생하지 않는다.

```
${data}
${dataArr[0]}
${dataMap["data"]}
```

JSTL

- JSTL은 JSP에서 Java 코드 블록(스크립트릿)을 사용하지 않고 HTML 형식을 유지하면서 조건문, 반복문, 간단한 연산 등 유용한 기능을 손쉽게 사용할 수 있도록 지원하기 위해 만들어진 표준 커스텀 태그 라이브러리이다.

- JSTL 라이브러리를 구현한 여러 버전이 있지만, Apache Standard Taglib을 주로 사용한다.

- 아래 링크에 접속해서 Impl(taglibs-standard-impl-1.2.5.jar)과 Spec(taglibs-standard-spec-1.2.5.jar)을 저장하자.

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

- 다운로드한 라이브러리 파일을 webapp/WEB-INF/lib 폴더에 복사해 넣어준다.

- JSTL을 JSP에서 사용하기 위해서는 taglib 지시어를 추가해야 한다.

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

JSTL - <c:set>

```
<c:set var="greeting" value="안녕하세요" />
<c:set var="title" value="<h1>JSP</h1>" />
<c:set var="numArr" value="${[1,2,3,4,5]}" />
```

JSTL - <c:if>

```
request.setAttribute("user", new User("최인규"));
RequestDispatcher dispatcher = request.getRequestDispatcher("index.jsp");
dispatcher.forward(request, response);
```

```
<c:if test="${user != null}">
    <p>Welcome, ${user.name}!</p>
</c:if>

<c:if test="${user == null}">
    <p>Please log in.</p>
</c:if>
```

JSTL - <c:choose>, <c:when>, <c:otherwise>

```
request.setAttribute("user", new User("최인규"));
RequestDispatcher dispatcher = request.getRequestDispatcher("index.jsp");
dispatcher.forward(request, response);
```

```
<c:choose>
    <c:when test="${user != null}">
        <p>${greeting}, ${user.name}!</p>
    </c:when>
    <c:otherwise>
        <p>${greeting}, Please log in.</p>
    </c:otherwise>
</c:choose>
```

JSTL - <c:forEach>

```
ArrayList<User> userList = new ArrayList<User>();
userList.add(new User("울버린"));
userList.add(new User("데드풀"));

request.setAttribute("userList", userList);
RequestDispatcher dispatcher = request.getRequestDispatcher("index.jsp");
dispatcher.forward(request, response);
```

```
<c:forEach var="u" items="${userList}">
    <p>${u.name}</p>
</c:forEach>
```