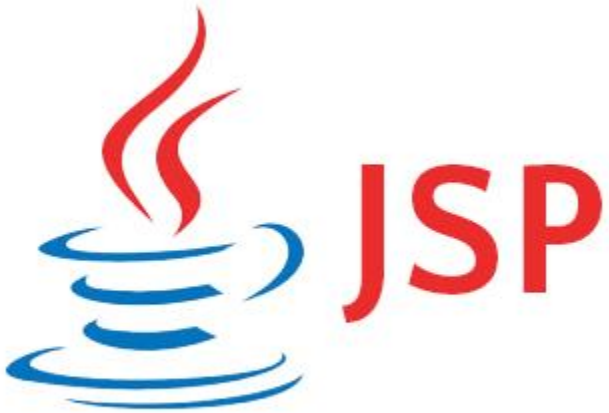
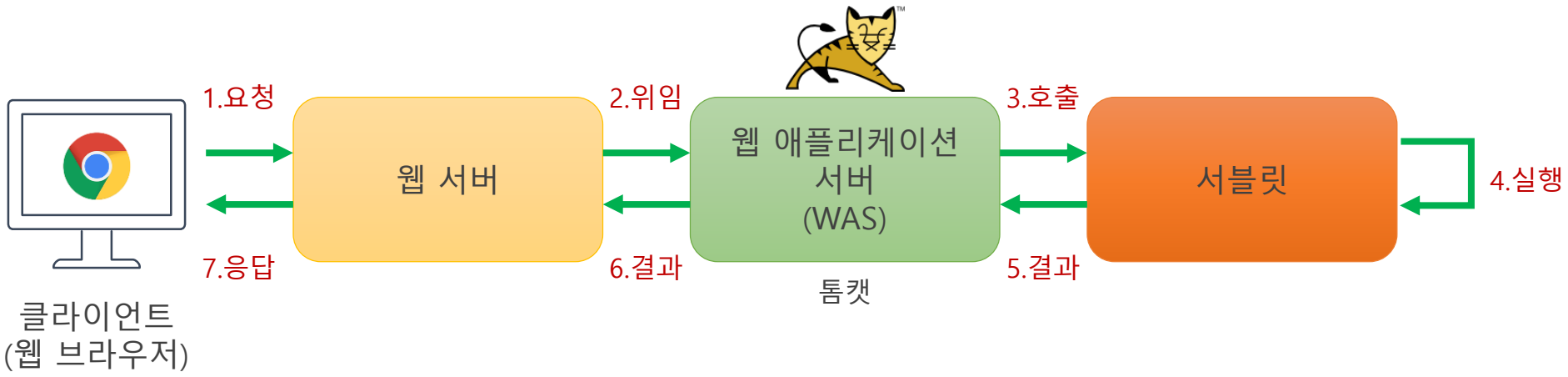


Servlet

Servlet

- Servlet
  - ✓ 클라이언트의 요청에 따라 동적으로 서비스를 제공하는 자바 클래스
  - ✓ 서버 쪽에서 실행(Server Side)
  - ✓ 일반 자바 클래스와 달리 단독적으로 실행되지 못하고 톰캣(Tomcat)과 같은 JSP/Servlet 컨테이너에서 실행됨 (main() 메소드 없어도 실행)
  - ✓ 자바 클래스 내부에서 HTML 문서를 만들어서 클라이언트에게 응답할 수 있음

- Servlet 동작 과정



Tomcat

- Tomcat
  - ✓ Apache Software Foundation(아파치 소프트웨어 재단)에서 개발한 Java 기반의 서블릿 컨테이너 and 웹 서버
  - ✓ Tomcat은 수컷 고양이를 의미함
  - ✓ JSP/Spring을 이용한 웹 사이트 구축에서 거의 필수적으로 사용되는 웹 애플리케이션 서버(WAS)

- Tomcat 공식 홈페이지
  - ✓ https://tomcat.apache.org/

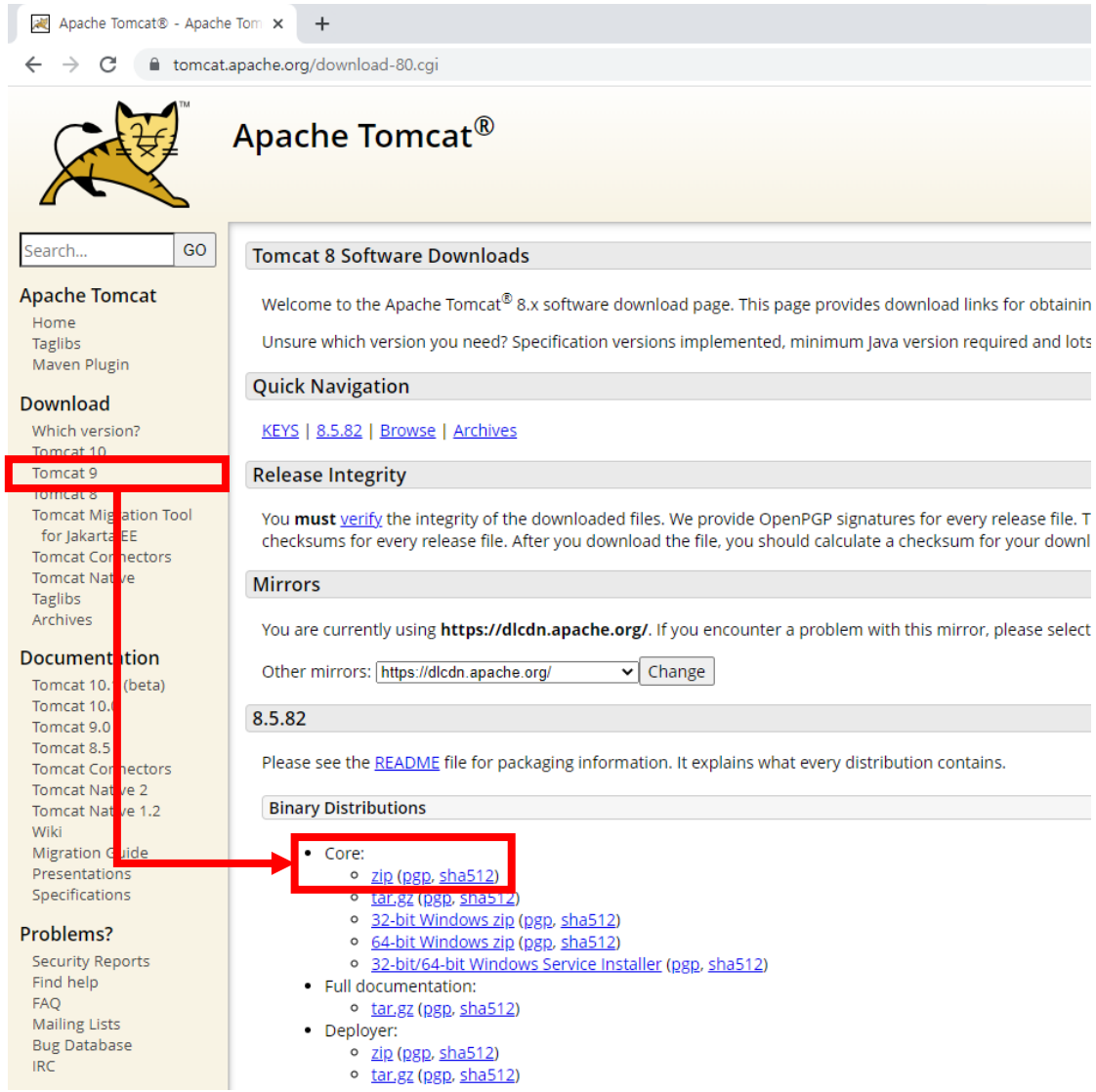
Tomcat 설치

Apache Tomcat 8.5

- Servlet 3.1
- JSP 2.3
- EL 3.0
- Java 7.0 이상

Apache Tomcat 9.0

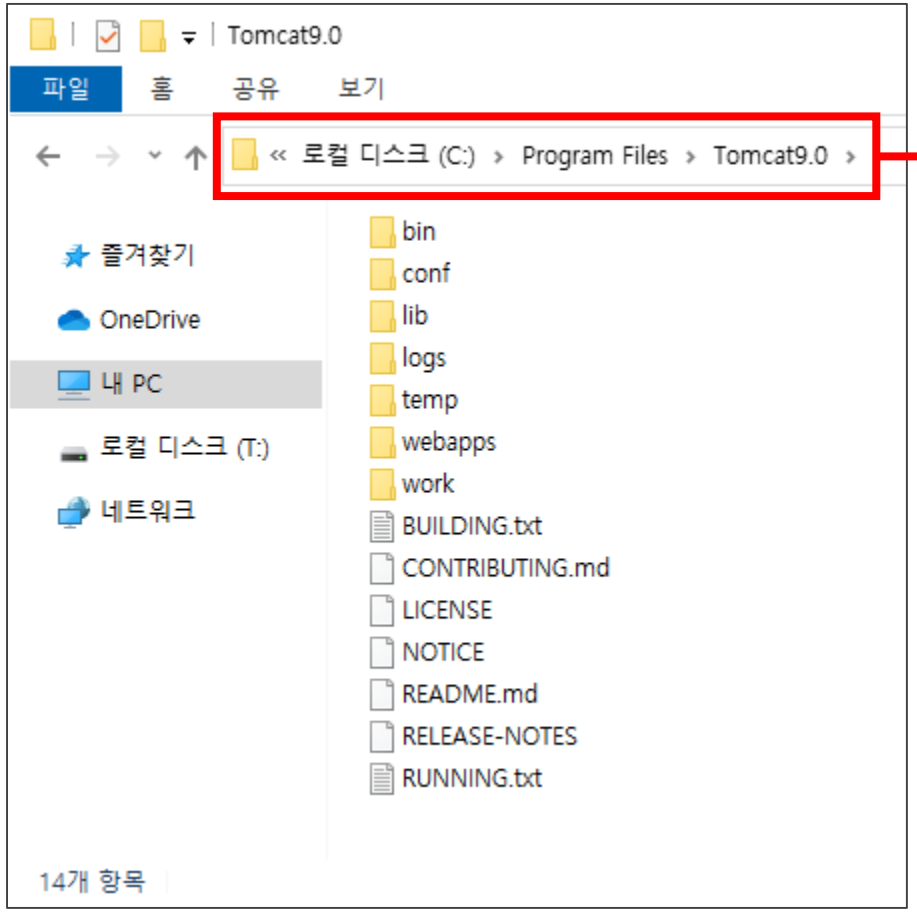
- Servlet 4.0
- JSP 2.3
- EL 3.0
- Java 8.0 이상

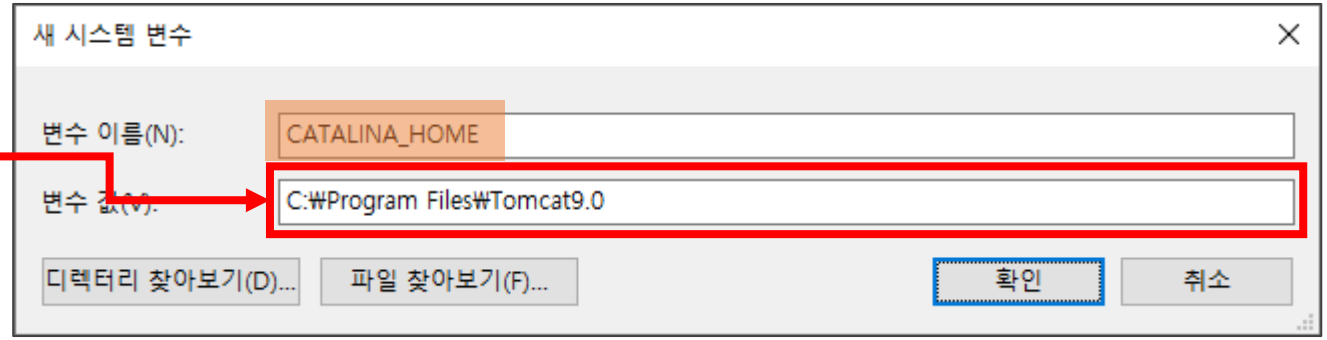


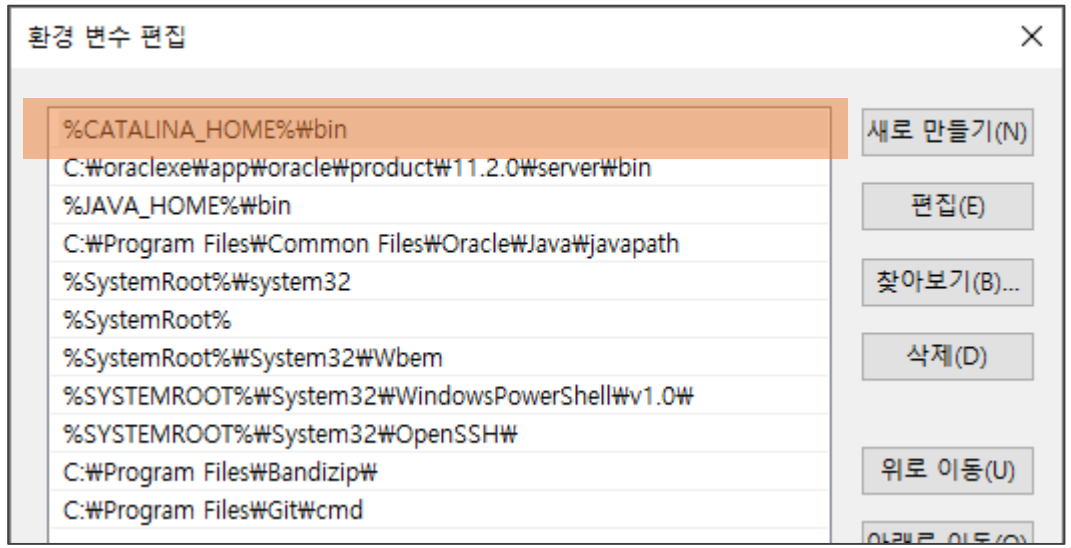
JSP

Tomcat 설치

- 다운로드 받은 압축 파일을 임의의 경로에 압축 해제한 뒤 CATALINA_HOME으로 환경변수 등록

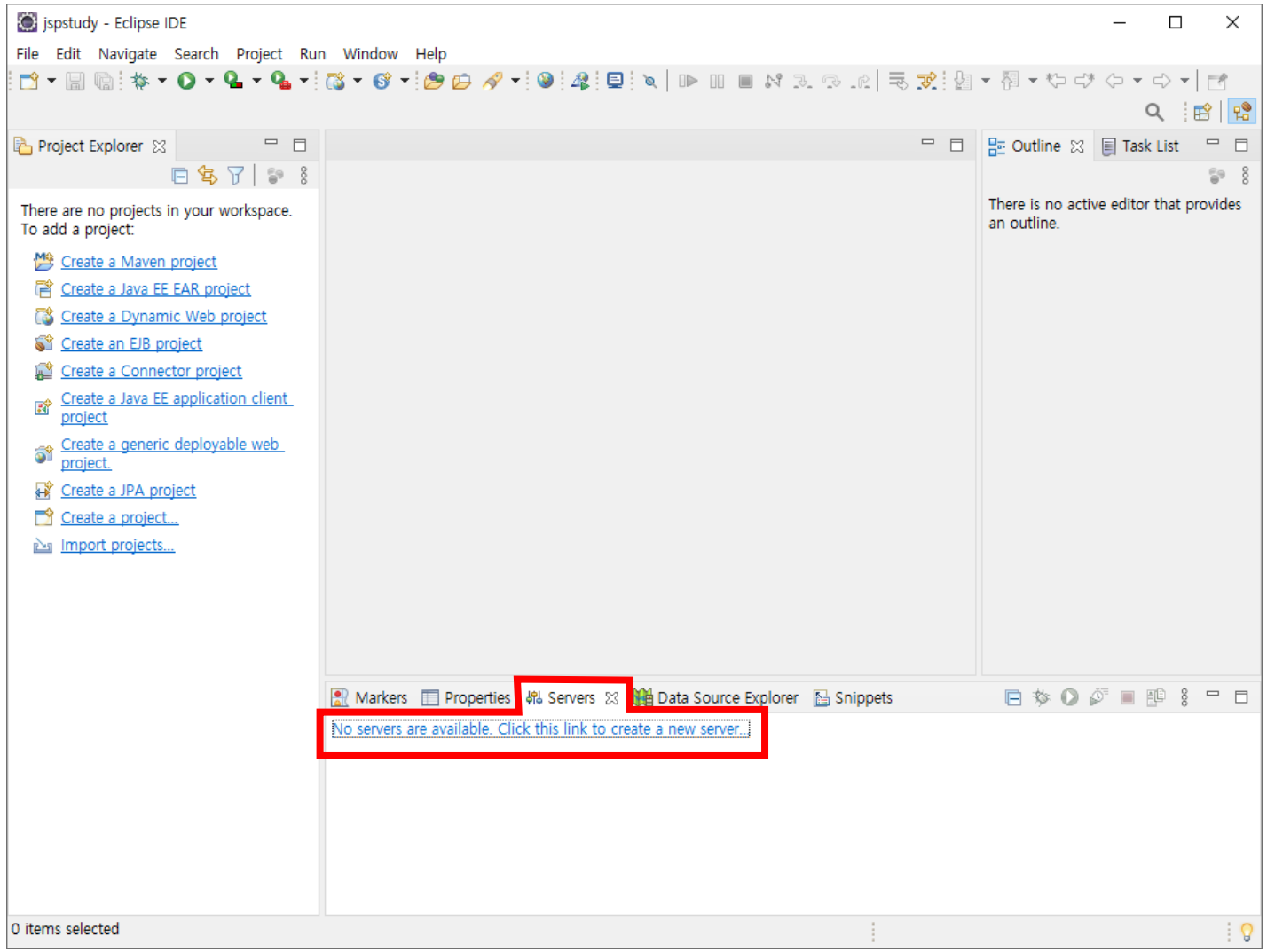




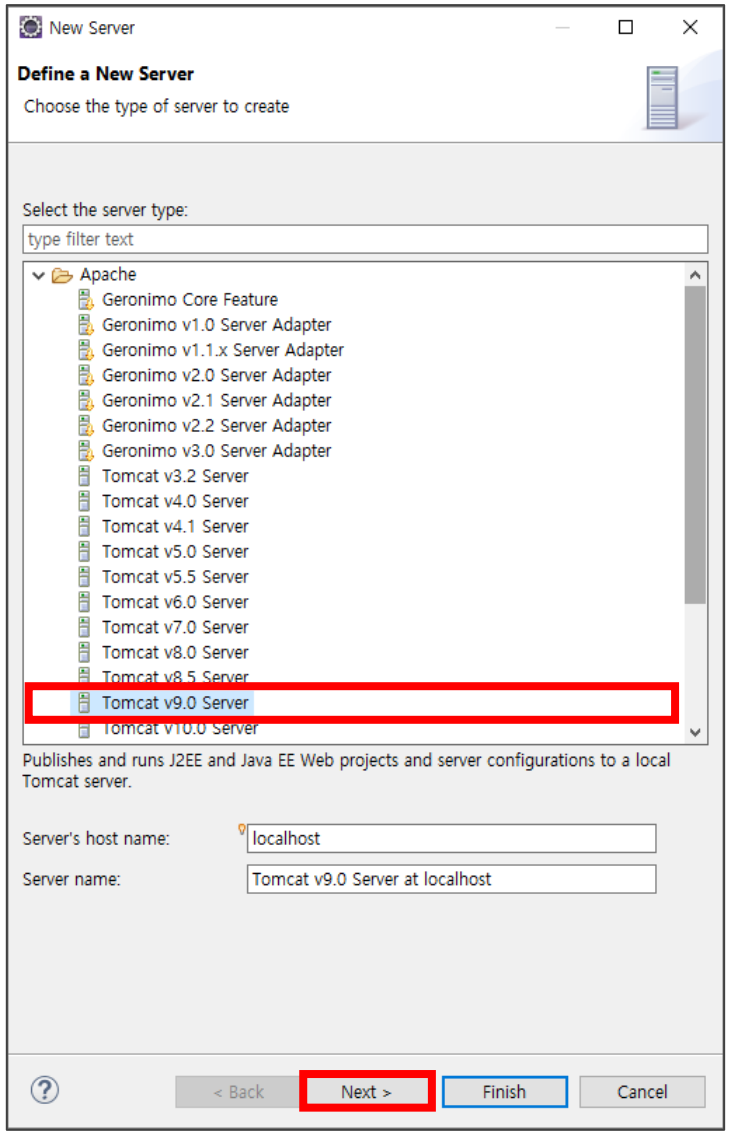


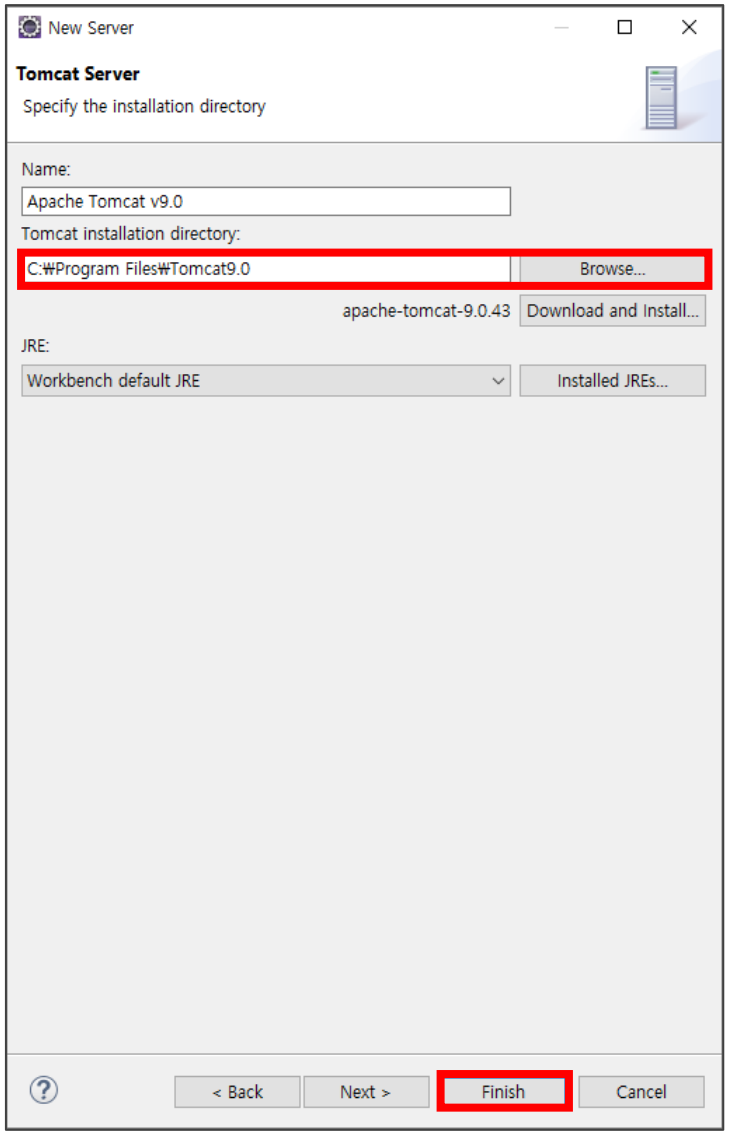
JSP

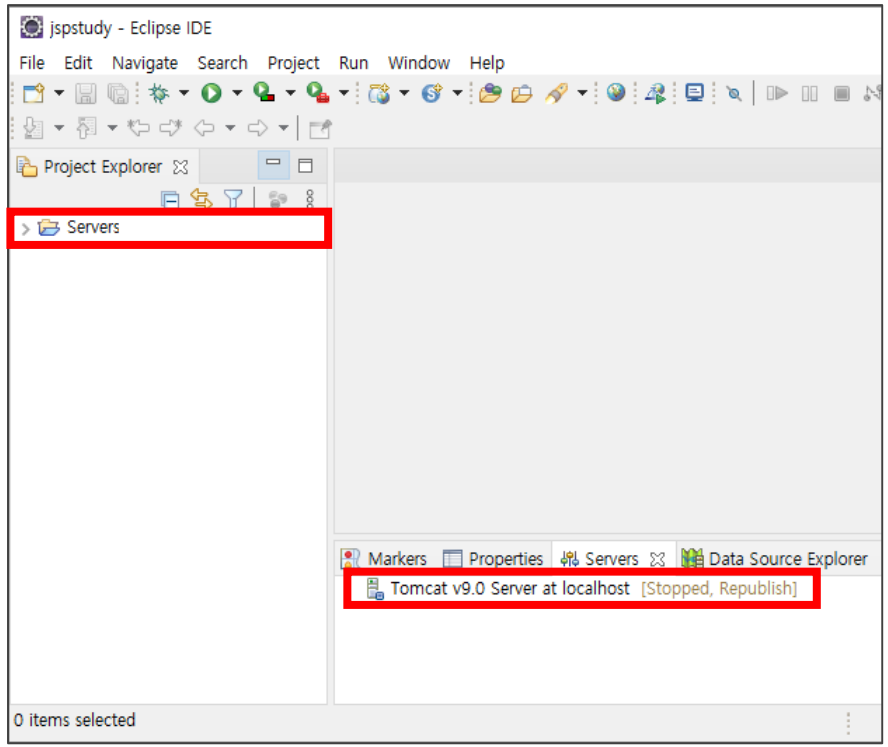
Eclipse에 Tomcat 등록



Eclipse에 Tomcat 등록

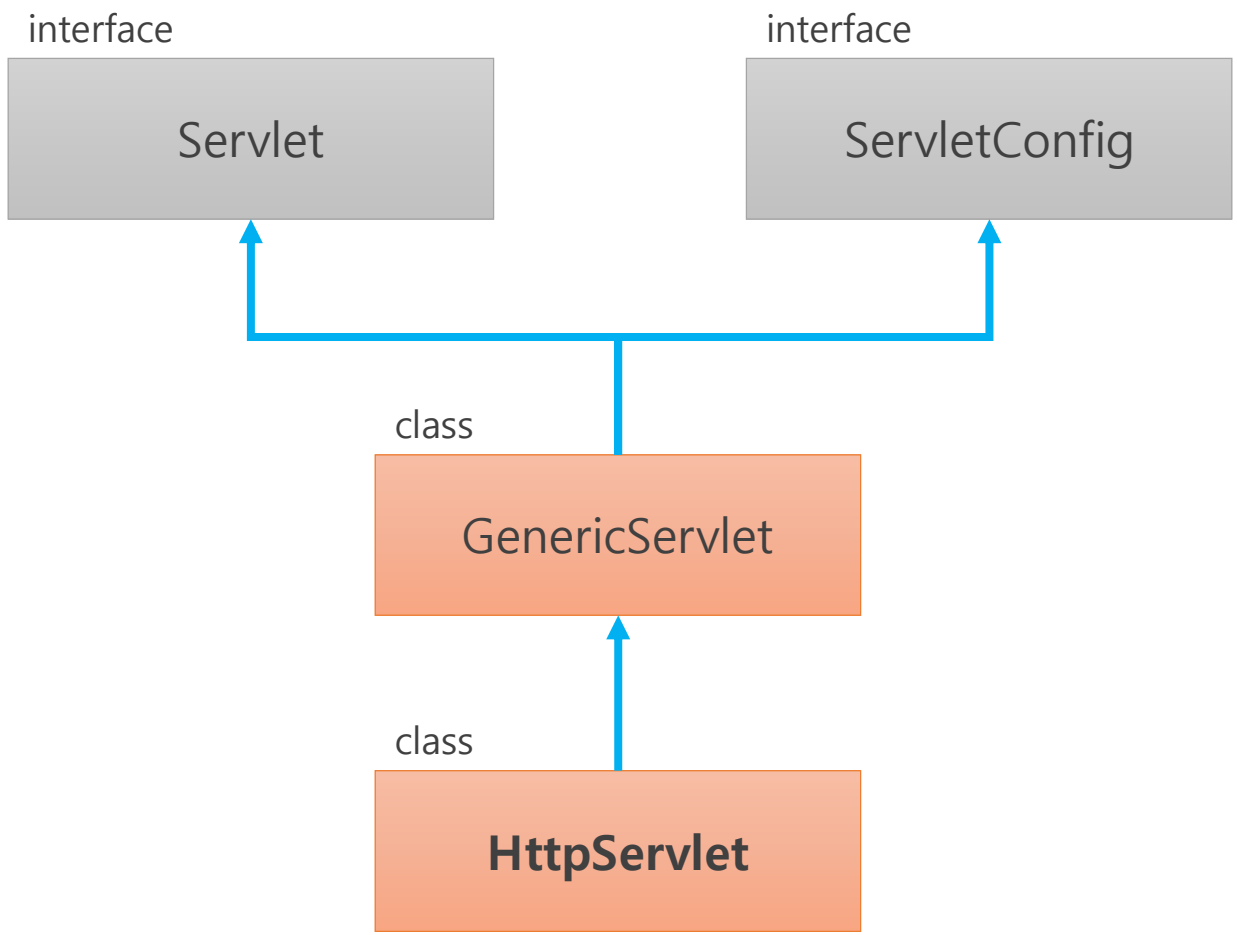






Servlet API

- Servlet 클래스 계층 구조



HttpServlet

- HttpServlet 클래스
  - ✓ 사용자 요청을 의미하는 HttpServletRequest 클래스의 슈퍼 클래스
  - ✓ 서버 응답을 의미하는 HttpServletResponse 클래스의 슈퍼 클래스

- HttpServlet 주요 메소드

메소드	역할
protected doDelete(HttpServletRequest request, HttpServletResponse response)	DELETE 요청(request)을 수행 service() 메소드를 통해서 호출
protected doGet(HttpServletRequest request, HttpServletResponse response)	GET 요청(request)을 수행 service() 메소드를 통해서 호출
protected doPost(HttpServletRequest request, HttpServletResponse response)	POST 요청(request)을 수행 service() 메소드를 통해서 호출
public service(ServletRequest request, ServletResponse response)	클라이언트의 요청(request)을 전달 받아 protected service() 메소드에 전달
protected service(ServletRequest request, ServletResponse response)	public service() 메소드로부터 요청(request)을 전달 받아 protected doXXX() 메소드에 전달

Servlet Life Cycle

- Servlet Life Cycle
  - ✓ 서블릿은 자바 클래스이므로 초기화/인스턴스생성/인스턴스소멸 등의 과정을 거침
  - ✓ 각 과정별로 미리 약속된 메소드가 호출되는 콜백 메소드가 존재함

- Life Cycle 주요 메소드

메소드	역할
public void init(ServletConfig config)	• 서블릿 요청 시 맨 처음 한 번만 호출 • 서블릿 생성 시 초기화 작업을 수행
protected service(HttpServletRequest request, HttpServletResponse response)	• 서블릿 요청 시 매번 호출 • post 방식의 요청이라면 doPost() 메소드를 호출하고 get 방식의 요청이라면 doGet() 메소드를 호출함
protected doGet(HttpServletRequest request, HttpServletResponse response)	• 서블릿 요청 시 매번 호출 • get 방식의 요청을 실제로 처리함
protected doPost(HttpServletRequest request, HttpServletResponse response)	• 서블릿 요청 시 매번 호출 • post 방식의 요청을 실제로 처리함
public void destroy()	• 서블릿이 메모리에서 소멸(unload)될 때 호출 • 서블릿이 사용하던 모든 자원의 해제(close)

필수

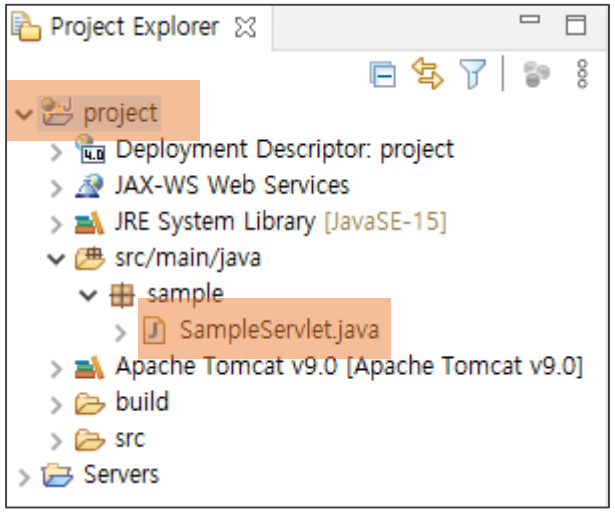
브라우저에서 Servlet 호출하기

- URL Mapping
  - ✓ 작성된 서블릿은 브라우저를 통해서 호출할 수 있음
  - ✓ 어느 서블릿이든 별명을 지정해 둘 수 있는데 이것을 URL Mapping이라고 함
  - ✓ URL Mapping은 기본적으로 서블릿의 이름과 같지만 다른 이름으로 수정할 수도 있음

- 브라우저에서 Servlet 호출하기
  - ✓ 브라우저를 열고 호출할 서블릿의 URL Mapping값을 포함한 주소를 입력함
  - ✓ 기본 URL 형식
    프로토콜://호스트:포트번호/ContextPath/URLMapping
  - ✓ 포트번호
    생략 가능, 기본값은 8080, 충돌 발생 시 Tomcat의 포트번호를 수정해서 사용
  - ✓ ContextPath
    기본적으로 프로젝트 이름과 동일함
  - ✓ URL Mapping
    기본적으로 서블릿 이름과 동일함

Servlet 호출 예시

- Servlet 호출 예시
  - ✓ 프로젝트명 : project
  - ✓ 패키지명 : sample
  - ✓ 서블릿명 : SampleServlet



http://localhost:8080/project/SampleServlet

URL Mapping 변경

- URL Mapping 변경 방법
  - ① Servlet 클래스에서 @WebServlet 애너테이션 등록하기
  - ② web.xml 파일에 <servlet> 태그 등록하기

1. @WebServlet 애너테이션

```
@WebServlet("/aaa")
public class SampleServlet extends HttpServlet {
```

http://localhost:8080/project/aaa

2. web.xml

```
<servlet>
  <servlet-name>SS</servlet-name>
  <servlet-class>sample.SampleServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>SS</servlet-name>
  <url-pattern>/aaa</url-pattern>
</servlet-mapping>
```

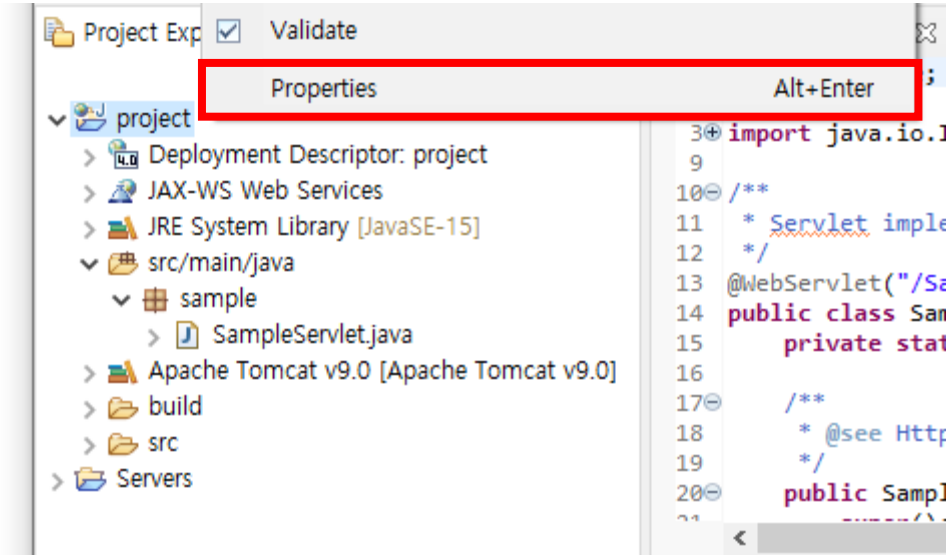
서블릿 별명 : SS
서블릿 원래 이름 : sample.SampleServlet

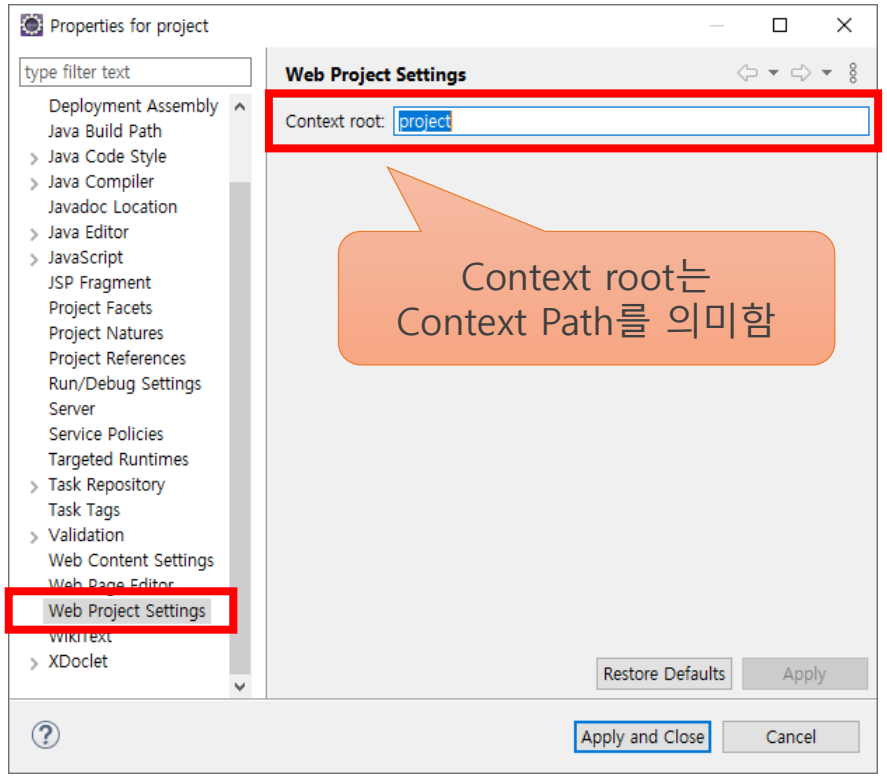
서블릿 별명 : SS
URL Mapping : /aaa

Context Path

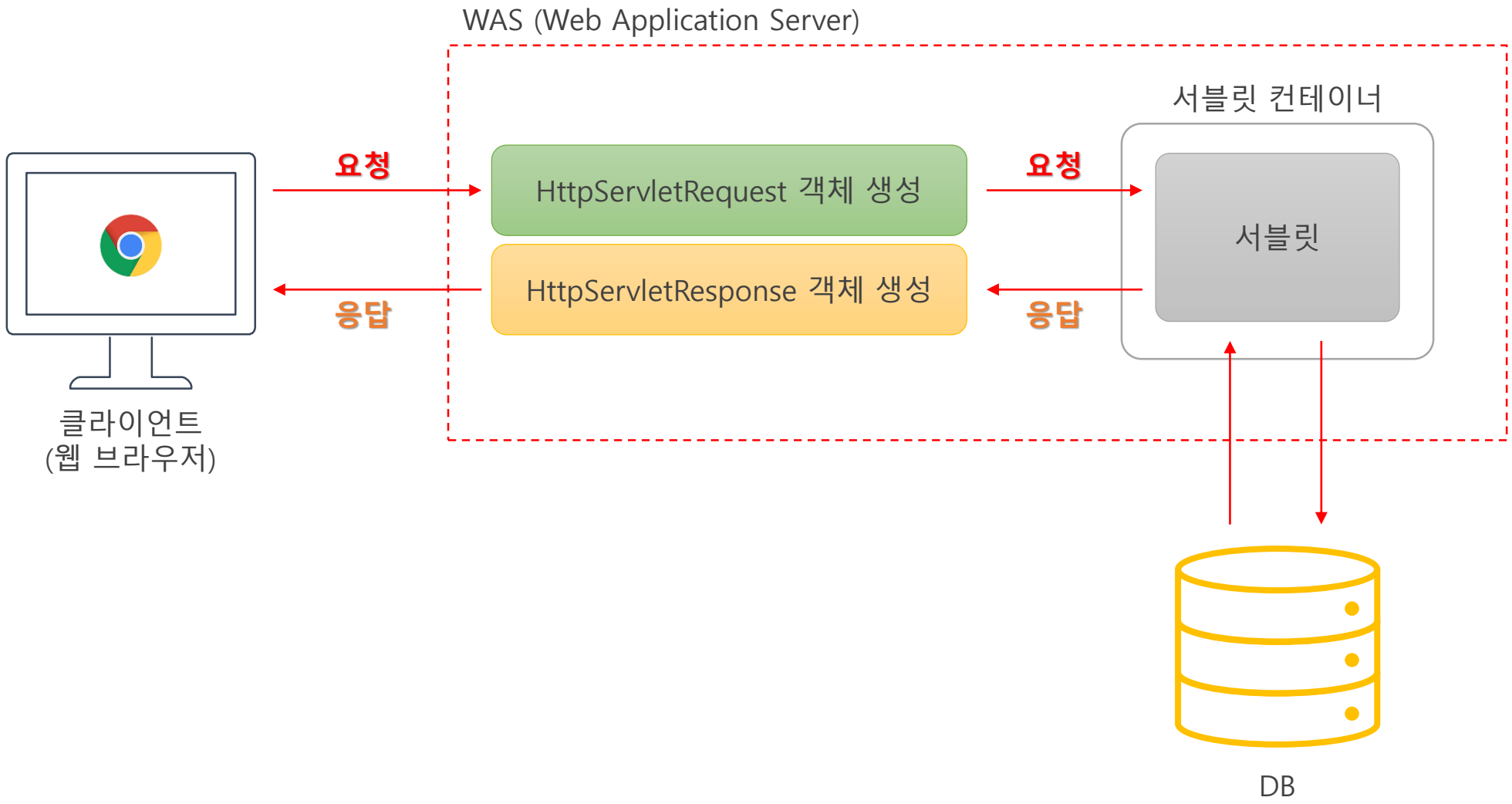
- Context Path
  - ✓ WAS에서 웹 애플리케이션을 구분하기 위한 기본 경로
  - ✓ 기본적으로 프로젝트이름과 컨텍스트패스는 동일함

- Context Path 변경 방법
  - ✓ 프로젝트 속성에서 컨텍스트패스를 변경할 수 있음





요청(request)과 응답(response)



HttpServletRequest

- HttpServletRequest 클래스
  - ✓ 클라이언트의 요청(request) 정보를 처리하는 클래스
  - ✓ WAS가 웹 브라우저로부터 서블릿 요청을 받으면 HttpServletRequest 객체가 생성됨
  - ✓ http 프로토콜의 request 정보를 서블릿에게 전달하기 위한 클래스
  - ✓ Header, Parameter, Cookie, URL 등의 정보를 처리할 수 있는 메소드를 포함하고 있음

- HttpServletRequest 주요 메소드

메소드	역할
public getContextPath()	요청한 컨텍스트를 가리키는 URI 반환
Cookie[] getCookies()	클라이언트가 현재 요청과 함께 보낸 쿠키 객체들을 배열로 반환
String getHeader(String name)	현재 요청에 포함된 특정 헤더 정보를 문자열로 반환
String getRequestURI()	요청한 URL의 컨텍스트와 파일 경로까지 모두 반환
HttpSession getSession()	현재 요청과 연관된 세션(Session)을 반환
String getParameter(String name)	현재 요청에 포함된 특정 파라미터 값을 문자열로 반환
String getParameterValues(String name)	현재 요청에 포함된 특정 파라미터 값을 문자열 배열로 반환

HttpServletResponse

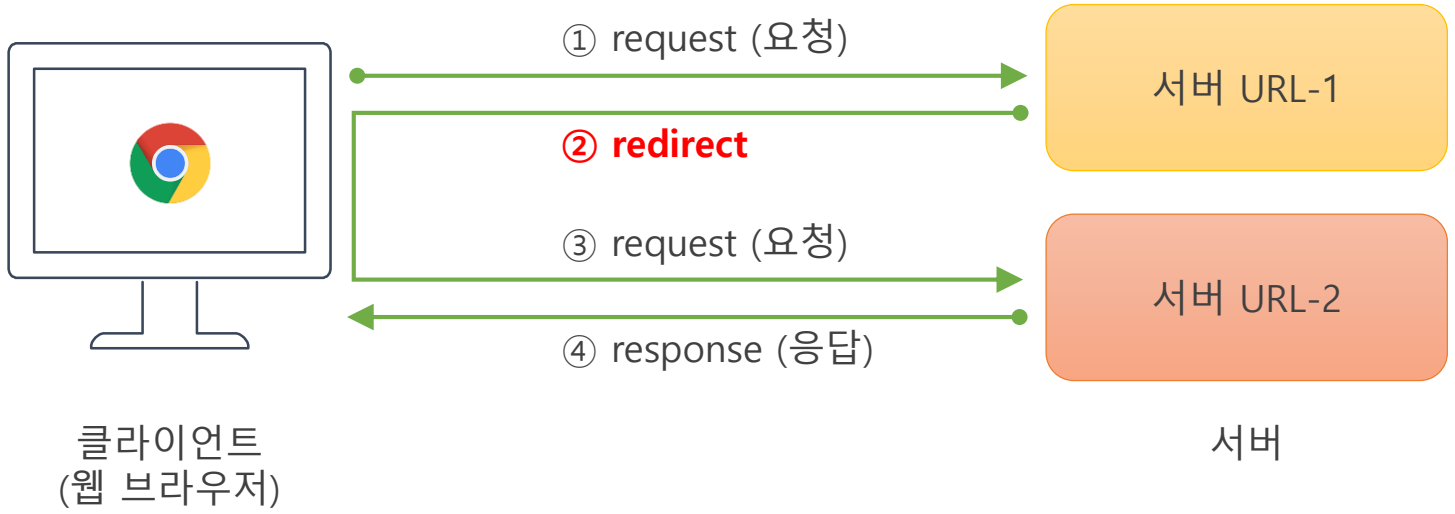
- HttpServletResponse 클래스
  - ✓ 서버의 응답(response) 정보를 처리하는 클래스
  - ✓ 응답을 위한 출력 스트림의 추출, 응답할 주소 처리, 응답할 데이터의 타입이나 문자셋 설정 등의 기능을 포함하고 있음

- HttpServletResponse 주요 메소드

메소드	역할
PrintWriter getWriter()	서버에서 클라이언트로 문자를 전송할 수 있는 출력 스트림 반환
void setContentType(String type)	응답 정보의 데이터 형식(MIME 형식)을 설정
void addCookie(Cookie cookie)	응답 정보에 쿠키를 추가(추가된 쿠키는 클라이언트에 저장됨)
void addHeader(String name, String value)	응답 정보에 특정 헤더 값을 저장
void sendRedirect(String location)	클라이언트에게 리다이렉트(Redirect)할 URL을 전송
void setStatus(int sc)	응답 정보에 상태코드를 저장

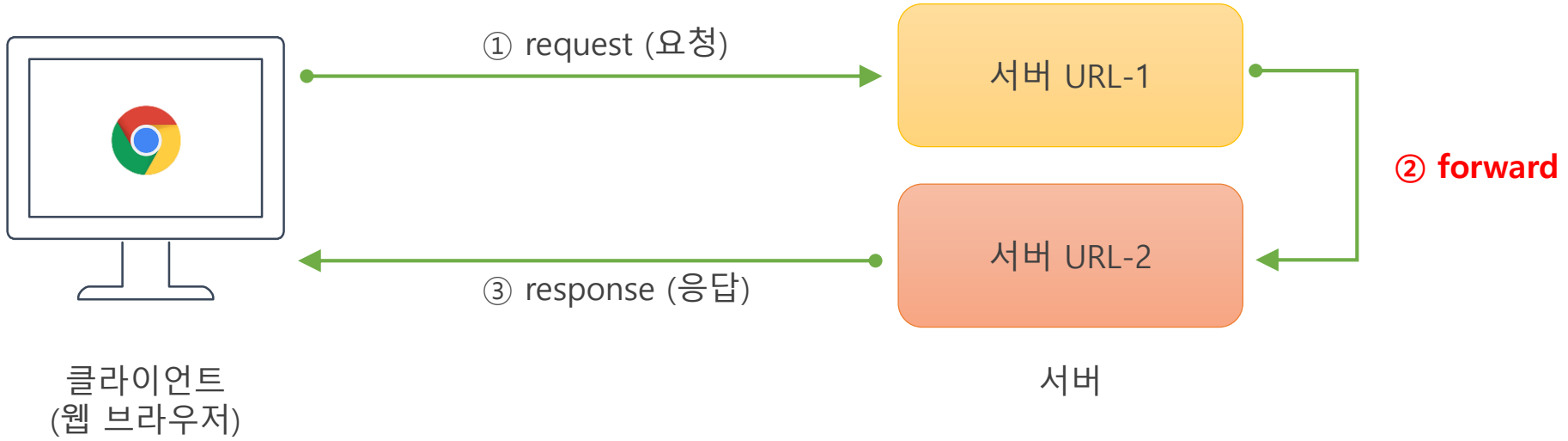
Redirect

- Redirect
  - ✓ 기존 request를 유지하지 않는 이동 방식
  - ✓ response를 이용해서 서버가 클라이언트에게 이동할 장소를 알려주는 방식
    response.sendRedirect(이동할경로)
  - ✓ 이동할 경로는 ContextPath를 포함한 전체 경로로 작성해야 함
  - ✓ 클라이언트가 직접 이동하는 방식이기 때문에 URL을 통해서 redirect 경로 확인이 가능함
  - ✓ DB가 변경되는 작업 이후에는 Redirect를 진행(INSERT, UPDATE, DELETE 이후에는 Redirect)



Forward

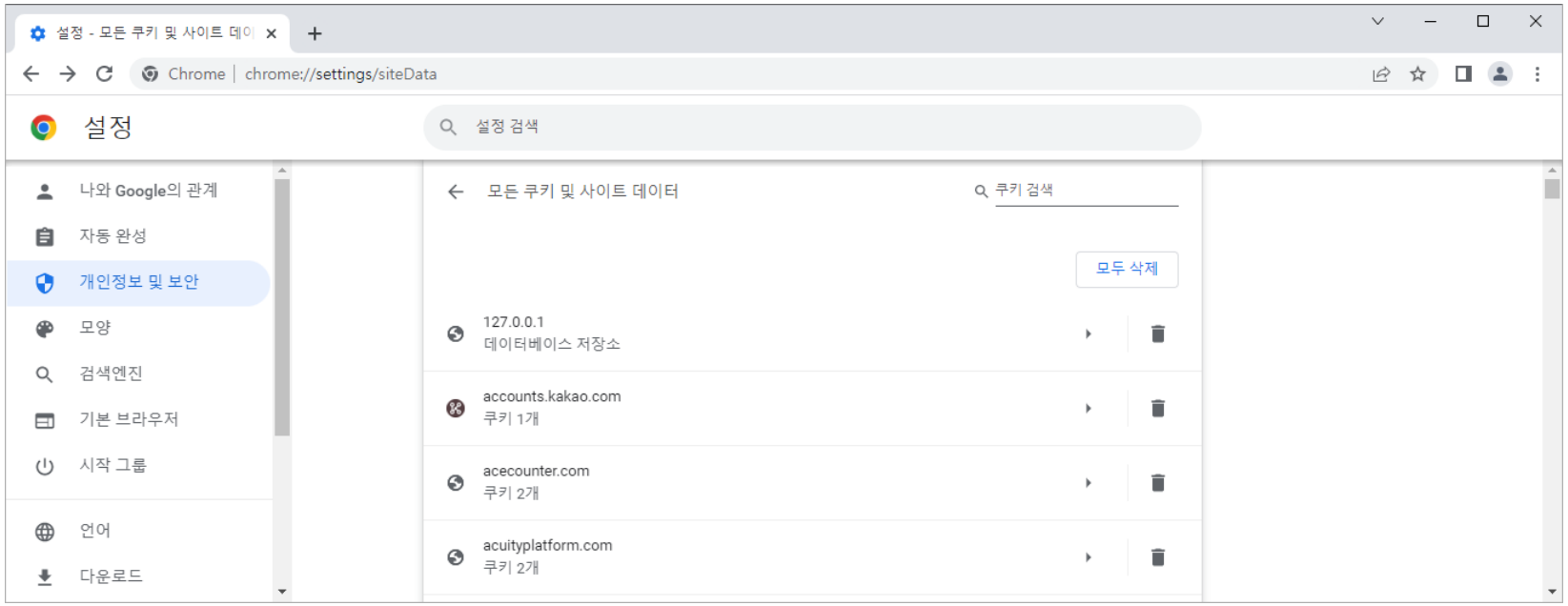
- Forward
  - ✓ 기존 request를 유지하는 이동 방식
  - ✓ 클라이언트의 요청(request)을 서버 내 다른 곳으로 이동하는 서버 내부의 이동 방식
    request.getRequestDispatcher(이동할경로).forward(request, response)
  - ✓ 이동할 경로는 ContextPath를 제외하고 서버 내부 경로만 작성해야 함
  - ✓ 서버 내부에서 이동하는 방식이기 때문에 클라이언트는 URL을 통해서 forward한 경로를 확인할 수 없음
  - ✓ DB 변경이 없는 작업이나 단순 이동의 경우 Forward를 진행(SELECT 이후에는 Forward)



Cookie

- Cookie
  - ✓ 클라이언트 측 PC에 저장되는 정보
    - ➢ 보안에 취약함
  - ✓ 웹 페이지들에서 참조해야 하는 공유 정보를 저장해 두고 사용하기 위해 쿠키를 활용할 수 있음
  - ✓ 4KB 용량 제한이 있음
  - ✓ 웹 브라우저에서 사용 유무를 설정할 수 있음
  - ✓ 웹 사이트 당 하나의 쿠키가 생성됨

오늘 더 이상 열지 않기 , 아이디 저장하기 등에서 쿠키가 활용됩니다.

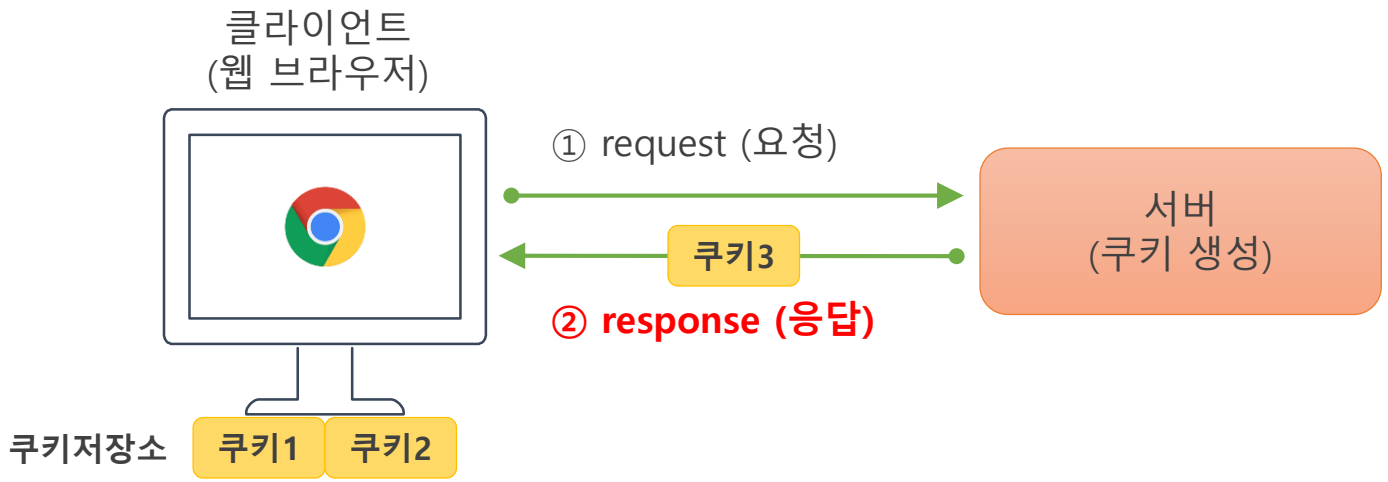


《크롬에서 쿠키 확인하기》
[설정] - [개인정보 및 보안] - [쿠키 및 기타 사이트 데이터] - [모든 쿠키 및 사이트 데이터 보기]

JSP

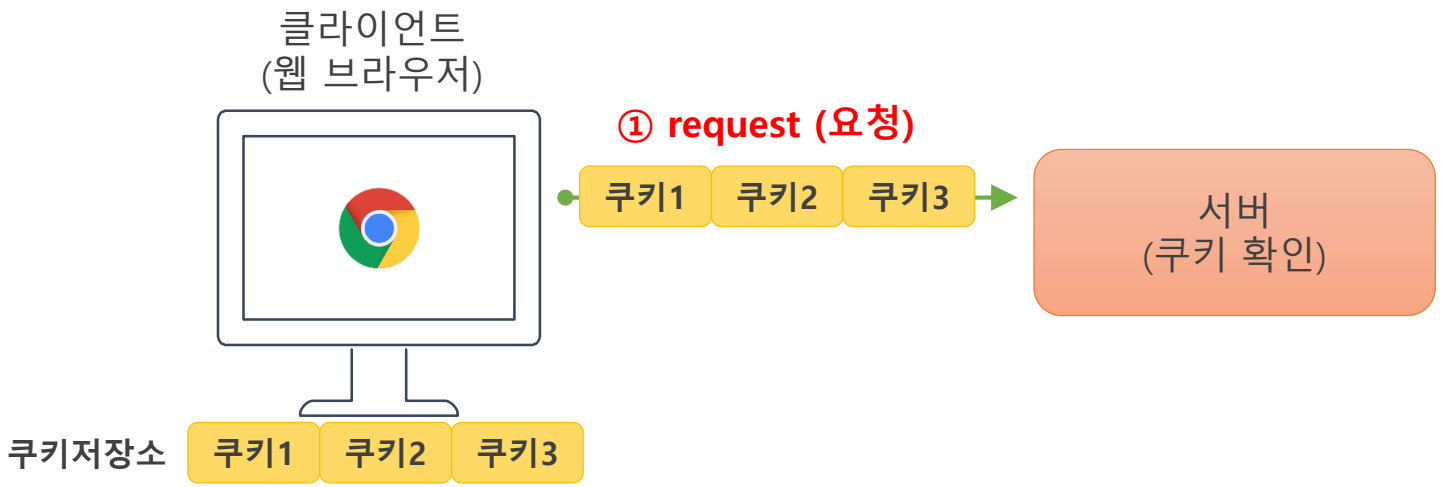
Cookie 저장

- Cookie 저장하기
  - ✓ 쿠키는 서버에서 생성함
  - ✓ 서버가 만든 쿠키를 클라이언트로 전달해서 클라이언트가 저장함
  - ✓ 서버가 만든 쿠키를 클라이언트로 전달하기 위해서 response를 사용함



Cookie 확인

- Cookie 확인하기
  - ✓ 쿠키는 클라이언트에 저장되어 있음
  - ✓ 클라이언트는 요청(request) 정보에 쿠키 정보를 담아서 서버에게 전달함
  - ✓ 클라이언트는 하나의 쿠키만 보낼 수는 없고, 모든 쿠키를 보내야 함

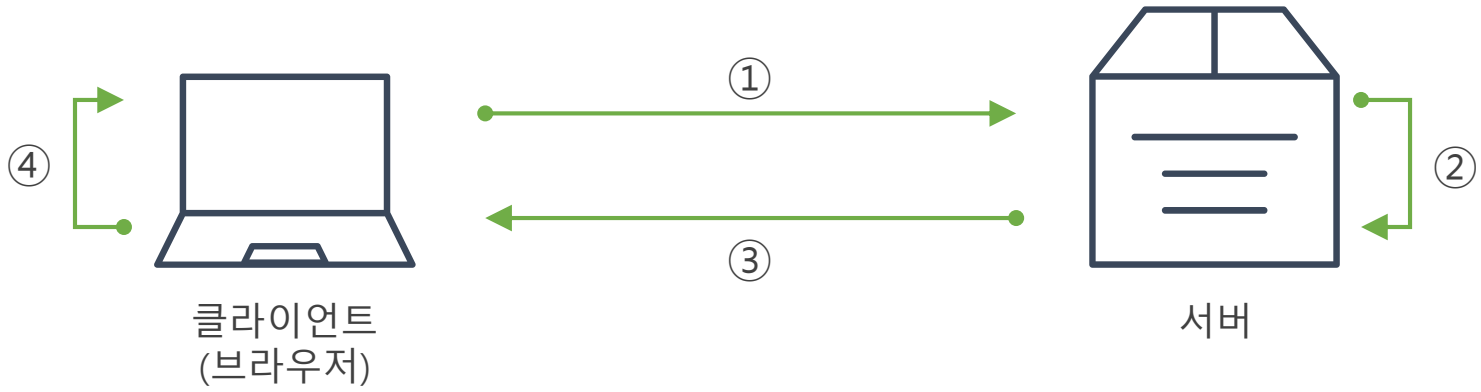


Session

- Session
  - ✓ 서버 메모리에 저장되는 정보
    - ➢ 보안에 유리함
  - ✓ 웹 페이지들에서 참조해야 하는 공유 정보를 저장해 두고 사용하기 위해 세션을 활용할 수 있음
  - ✓ 유효 시간을 가짐(기본 30분)
  - ✓ 브라우저 당 하나의 세션이 생성됨(세션ID를 통해서 각 세션을 구분함)

사용자 로그인 정보,
장바구니 등에서 세션이
활용됩니다.

- Session 실행 과정
  - ① 브라우저로 특정 사이트 접속
  - ② 접속한 브라우저의 세션 객체 생성
  - ③ 생성된 세션 객체의 ID를 브라우저에 응답
  - ④ 브라우저는 서버로부터 받은 세션 ID를 세션 쿠키 형태로 저장(쿠키명 : jsessionId)



Session 생성

- Session 생성
  - ✓ 세션은 HttpSession 클래스 객체임
  - ✓ 세션은 HttpServletRequest 클래스 객체의 getSession() 메소드를 호출해서 생성함
- HttpServletRequest 클래스의 Session 생성 메소드

메소드	역할
HttpSession getSession()	기존의 세션 객체가 존재하면 해당 세션 객체를 반환 없으면 새로 생성한 세션 객체를 반환
HttpSession getSession(boolean arg)	기존의 세션 객체가 존재하면 해당 세션 객체를 반환 없으면 새로 생성한 세션 객체를 반환 (true) 없으면 null을 반환 (false)

- HttpSession 주요 메소드

메소드	역할
long getCreateTime()	세션이 생성된 타임스탬프(Timestamp) 값 반환
String getId()	세션에 할당된 고유 식별자 반환
void invalidate()	현재 생성된 세션 소멸
void setMaxInactiveInterval(int interval)	세션 유지 시간을 초 단위로 설정

Servlet Attribute

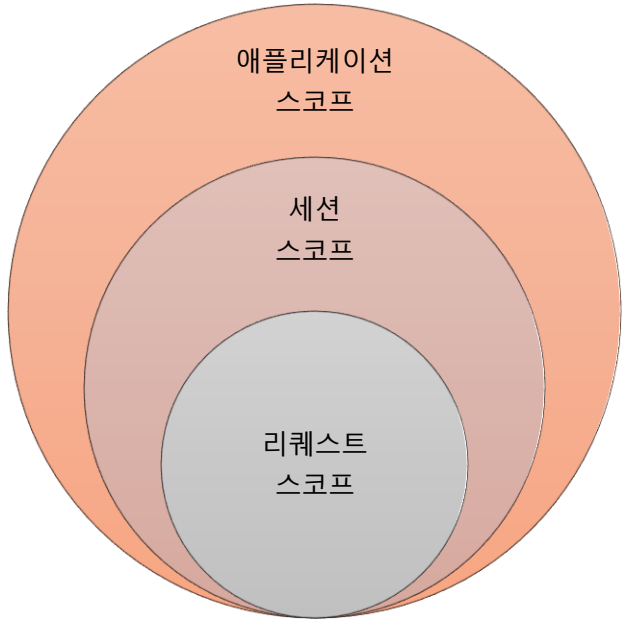
- Servlet Attribute
  - ✓ 서블릿 속성
  - ✓ 웹 프로그램 실행 시 데이터를 서블릿 관련 객체에 저장하는 방법
  - ✓ 속성으로 저장해 둔 데이터를 서블릿이나 JSP들이 공유해서 사용할 수 있음

- Attribute(속성) 관련 메소드

메소드	역할
void setAttribute(String name, Object obj)	지정한 name 속성으로 obj 저장
Object getAttribute(String name)	지정한 name 속성의 값 반환 Object를 반환하기 때문에 캐스팅이 필요할 수 있음
void removeAttribute(String name)	지정한 name 속성 삭제

Scope

- Scope
  - ✓ 서블릿 속성(Attribute)에 접근할 수 있는 접근 범위를 의미함
  - ✓ 스코프가 다른 경우에는 해당 속성에 접근할 수 없음
  - ✓ 동일한 이름의 속성을 서로 다른 스코프에 저장할 수 있음
  - ✓ 동일한 이름의 속성인 경우 다음의 우선 순위를 가짐
    - ➢ 리퀘스트 스코프 > 세션 스코프 > 애플리케이션 스코프
  - ✓ 스코프 별 저장 정보
    - ➢ 리퀘스트 스코프 : 단순 요청 정보
    - ➢ 세션 스코프 : 로그인 회원 정보, 장바구니 정보
    - ➢ 애플리케이션 스코프 : 총 방문자 수



- Scope 종류

스코프 종류	서블릿 API	접근 범위
애플리케이션 스코프	ServletContext	동일한 애플리케이션에서 접근 가능
세션 스코프	HttpSession	동일한 브라우저에서 접근 가능
리퀘스트 스코프	HttpServletRequest	하나의 요청/응답 사이클에서만 접근 가능