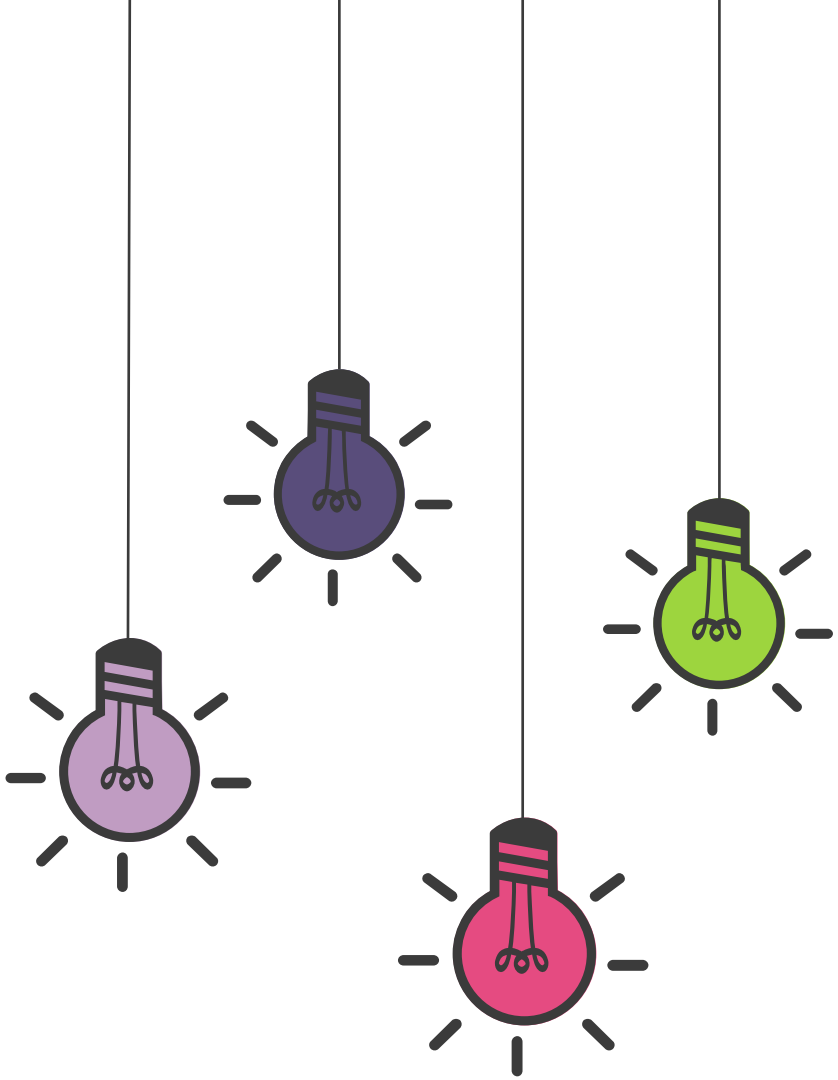


Servlet 프로그래밍

Servlet 프로그래밍

Index

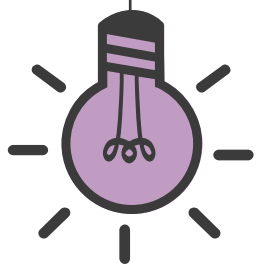
01 Servlet의 개요

02 Query 문
자열 처리

03 상태 정보 관리

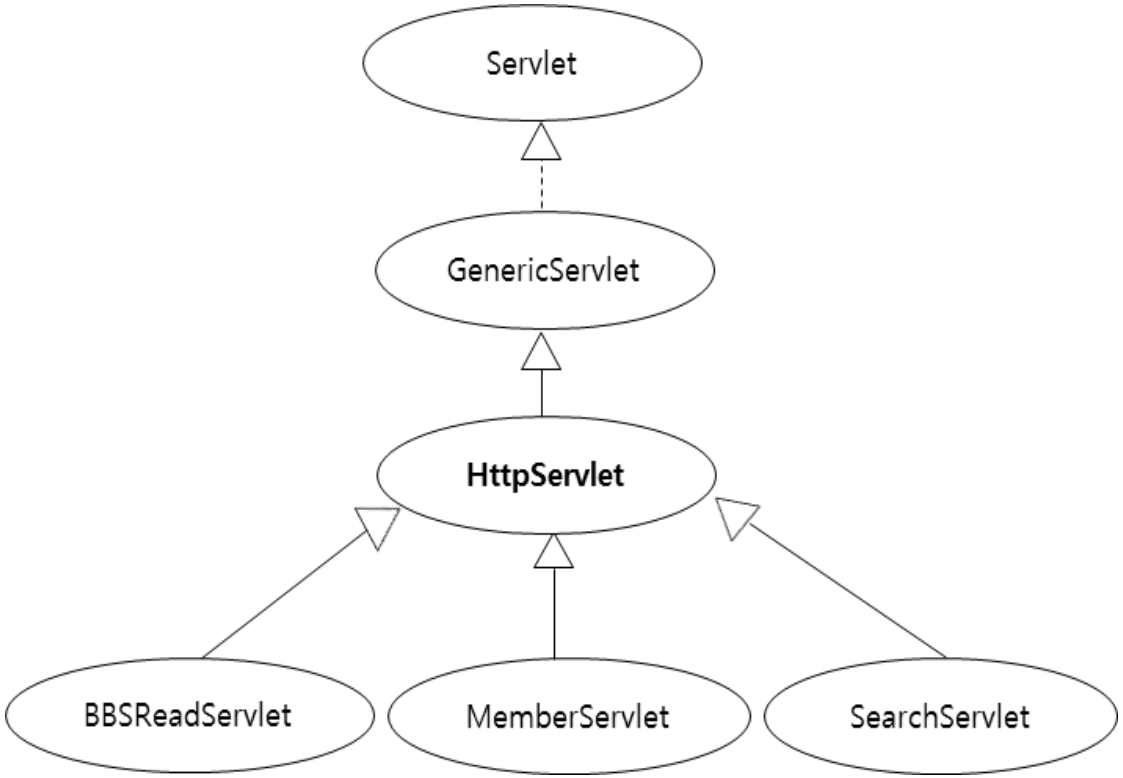
04 요청 재지정

01 Servlet의 개요



1-1. Servlet의 구현

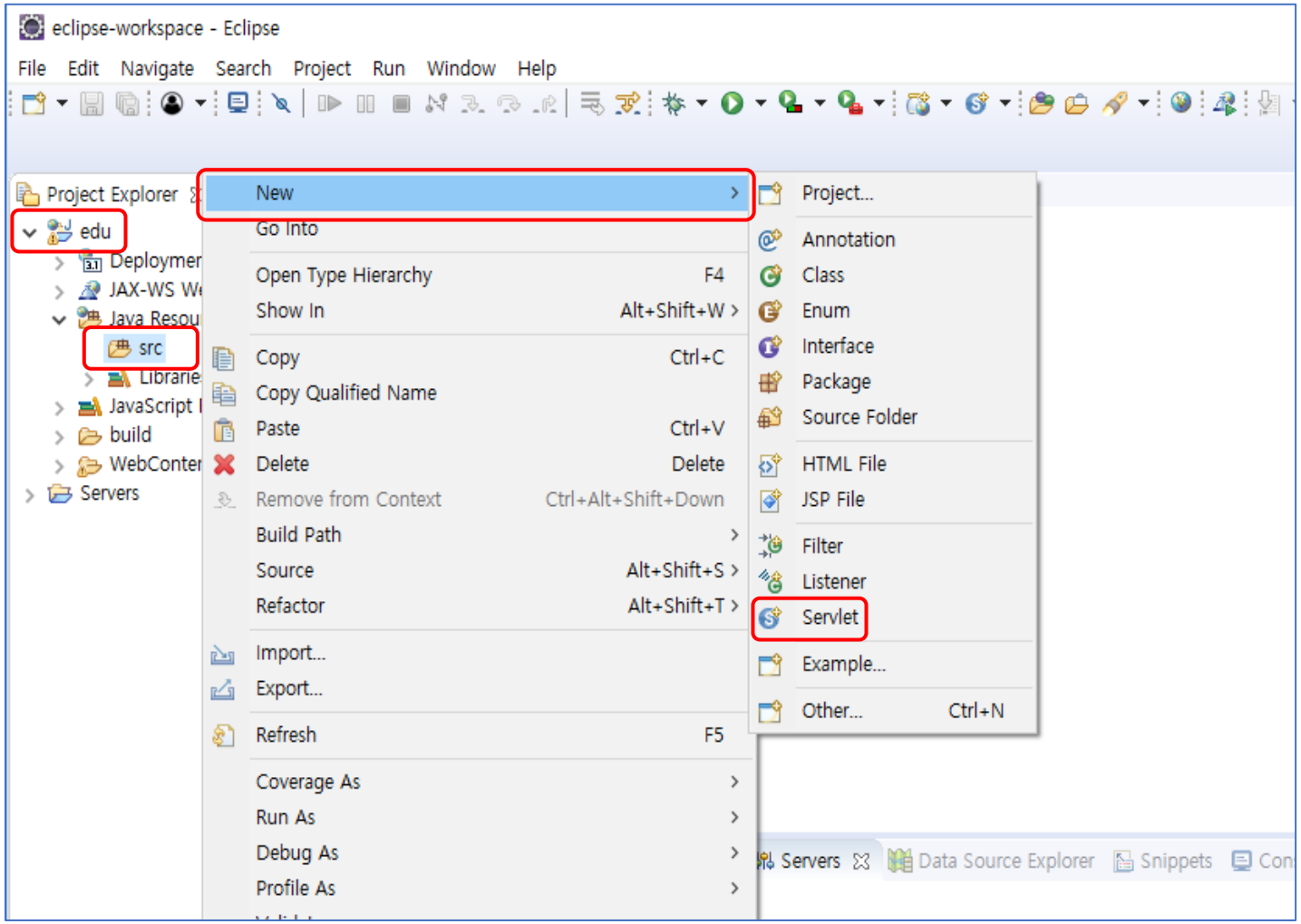
✓ Servlet 클래스를 생성할 때 구현하려는 기능과 관계없이 HttpServlet 클래스를 상속해야 함



Servlet의 상속 구조

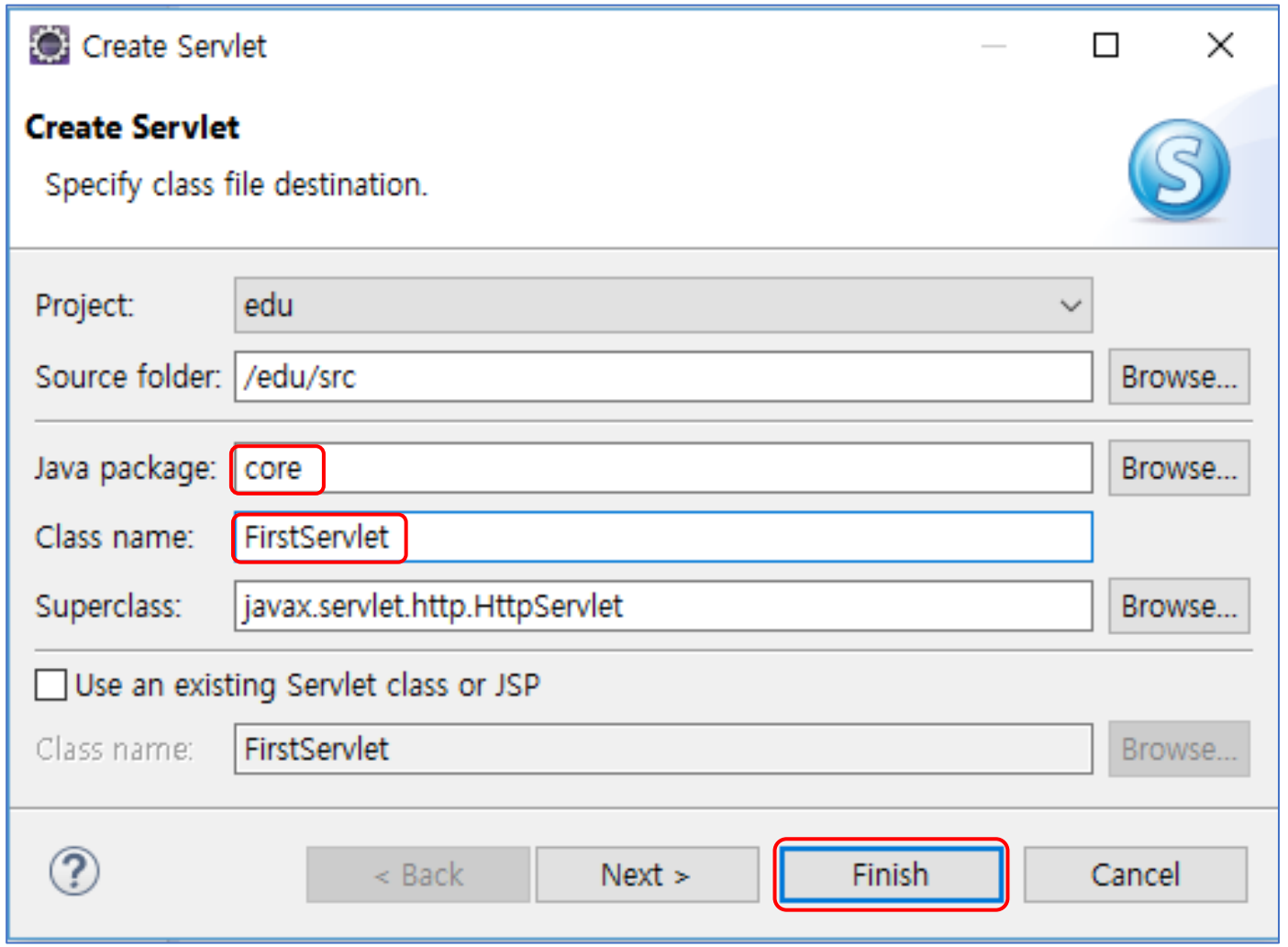
1-1. Servlet의 구현

- ✓ Eclipse를 시작한 다음, 화면 왼쪽의 Project Explorer 뷰에서 edu라는 dynamic web project를 오픈
- ✓ 하위 폴더에서 Java Resources의 src를 선택한 다음 마우스 오른쪽 버튼을 클릭하고 New > Servlet을 차례대로 선택



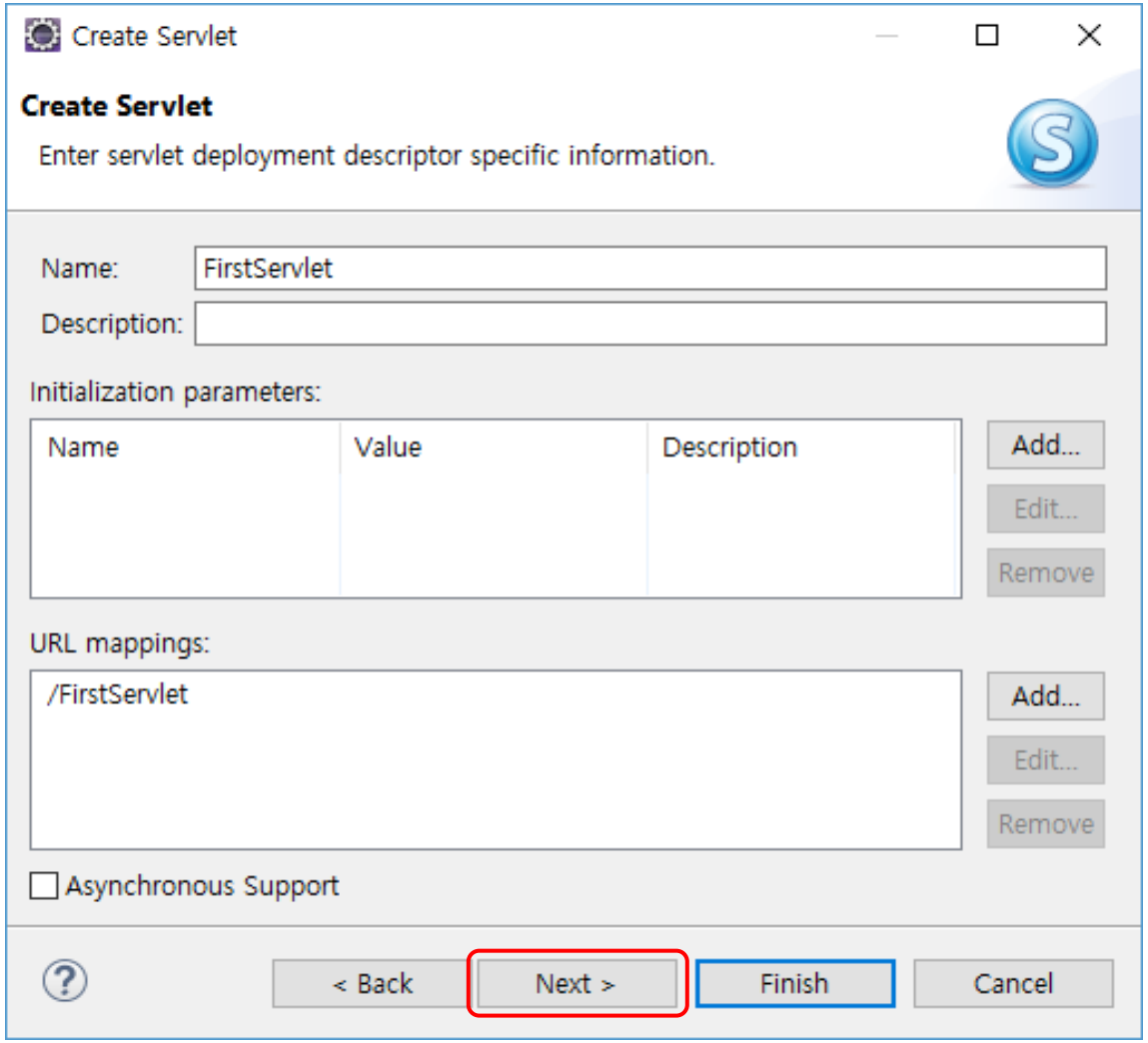
1-1. Servlet의 구현

✓ Create Servlet이라는 팝업창이 출력되면 Java package 항목에는 **core**, Class Name 항목에는 **FirstServlet**을 입력한 후 Next 버튼을 클릭



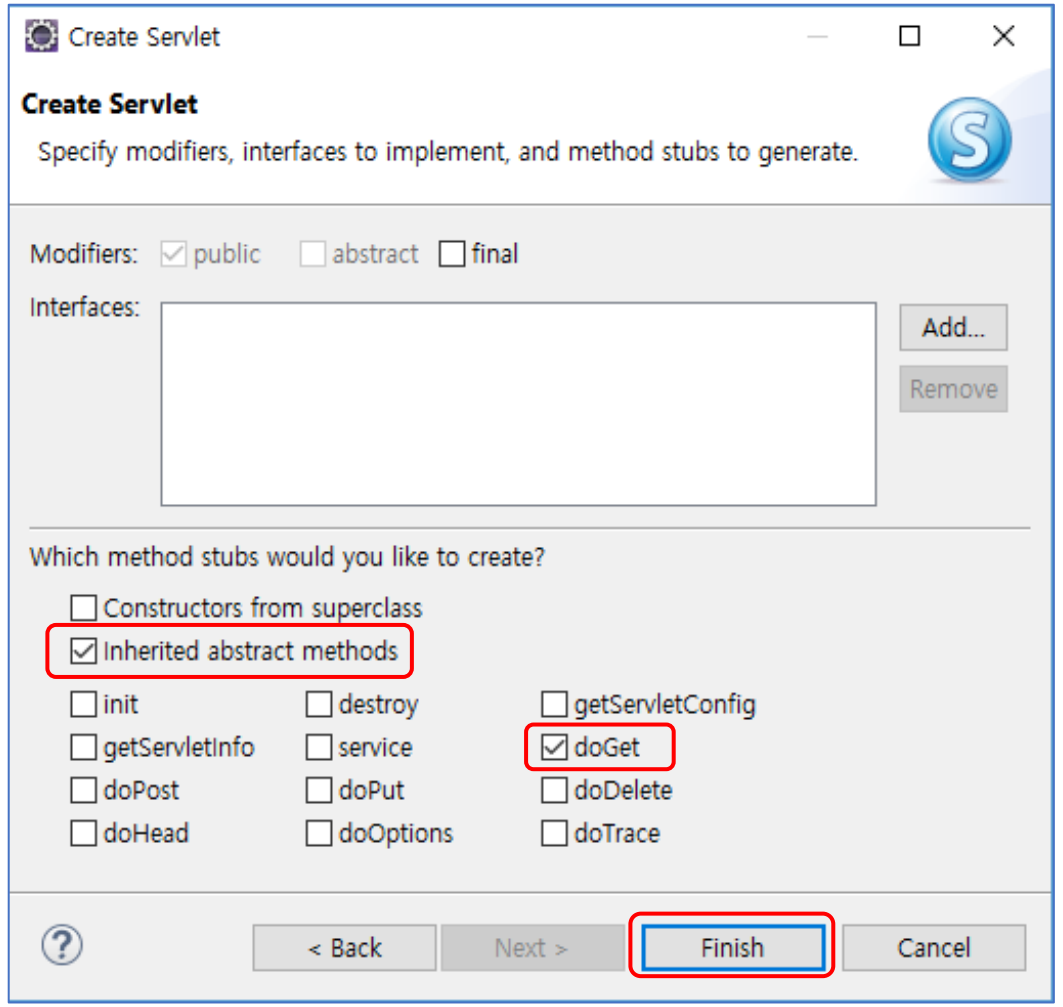
1-1. Servlet의 구현

✓ 여기에서는 특별히 수정하거나 입력할 내용이 없으므로 바로 Next 버튼을 클릭



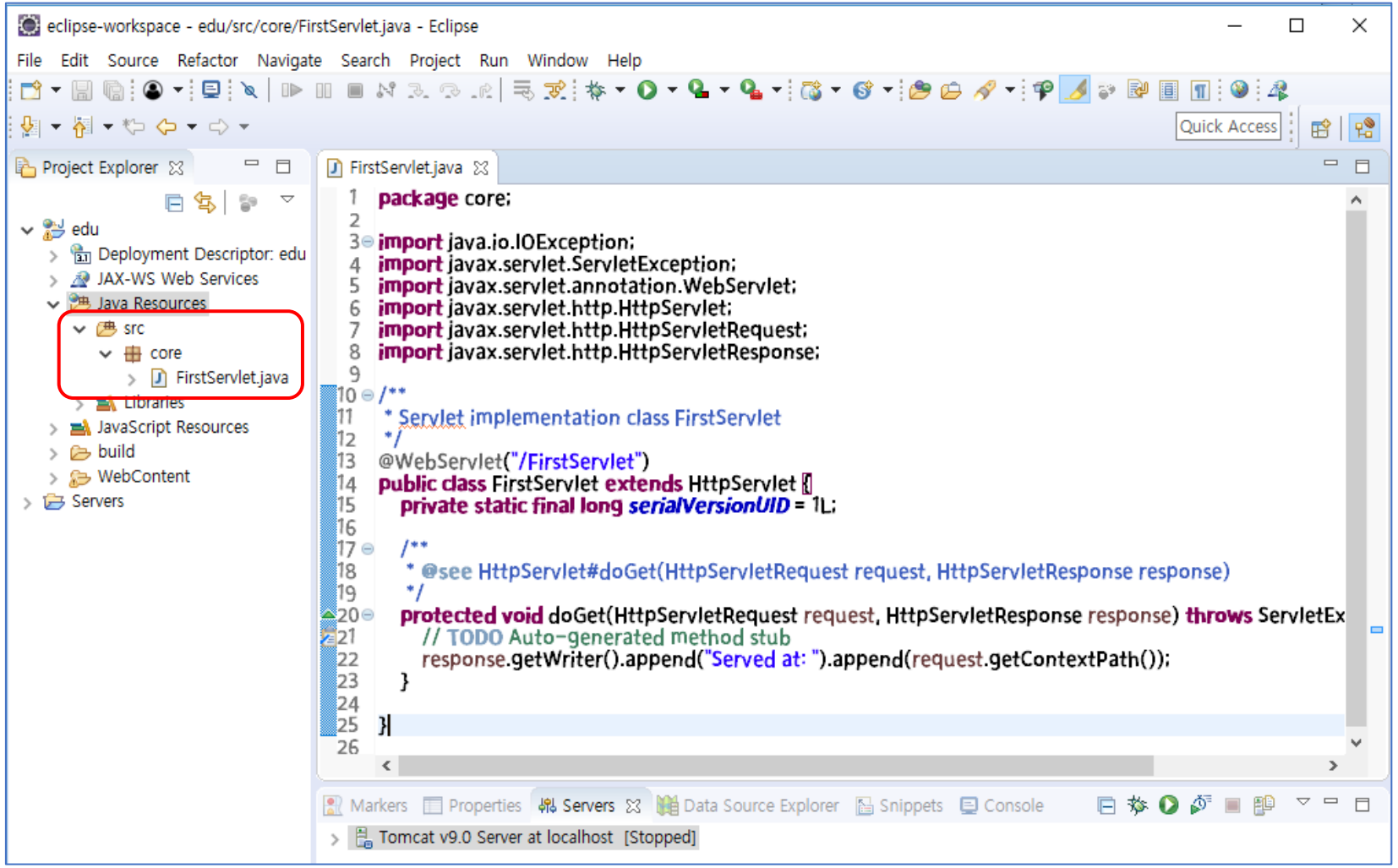
1-1. Servlet의 구현

✓ Inherited abstract methods와 doGet에 체크를 한 후 Finish 버튼을 클릭



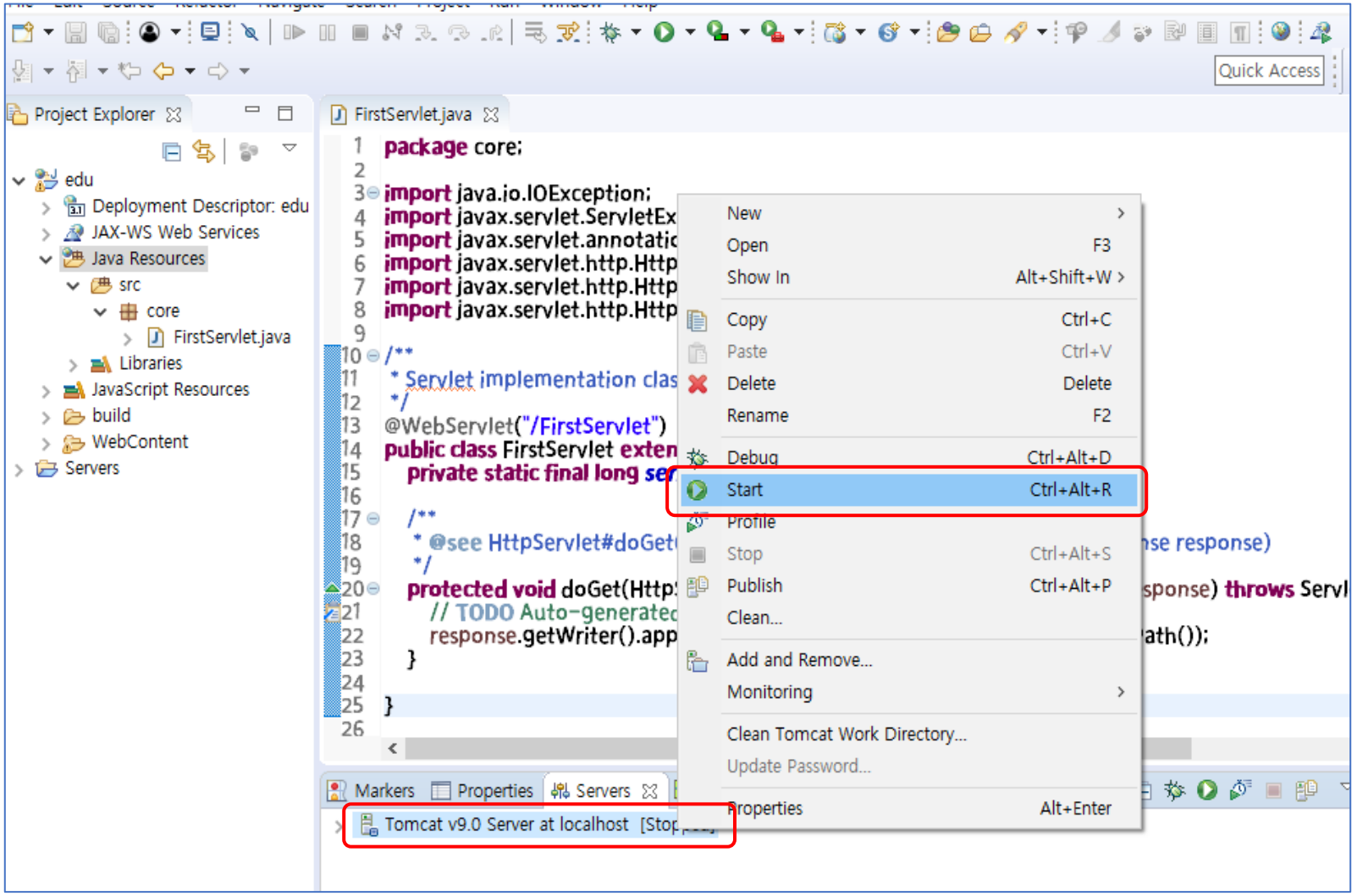
1-1. Servlet의 구현

- ✓ HttpServlet을 상속하고 있고 doGet( ) 메서드를 오버라이딩



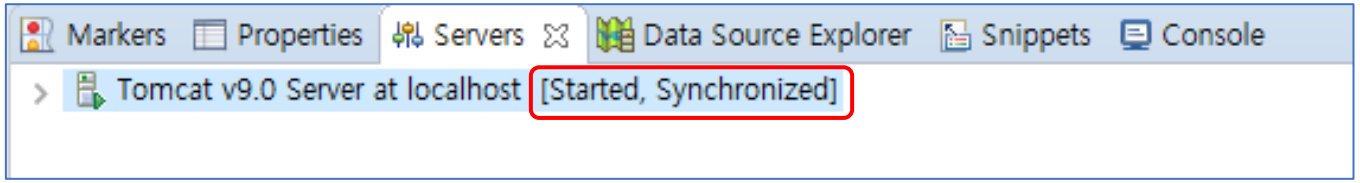
1-1. Servlet의 구현

✓ 오른쪽 하단 Server뷰의 Tomcat 서버를 선택하고 마우스 오른쪽 버튼을 클릭



1-1. Servlet의 구현

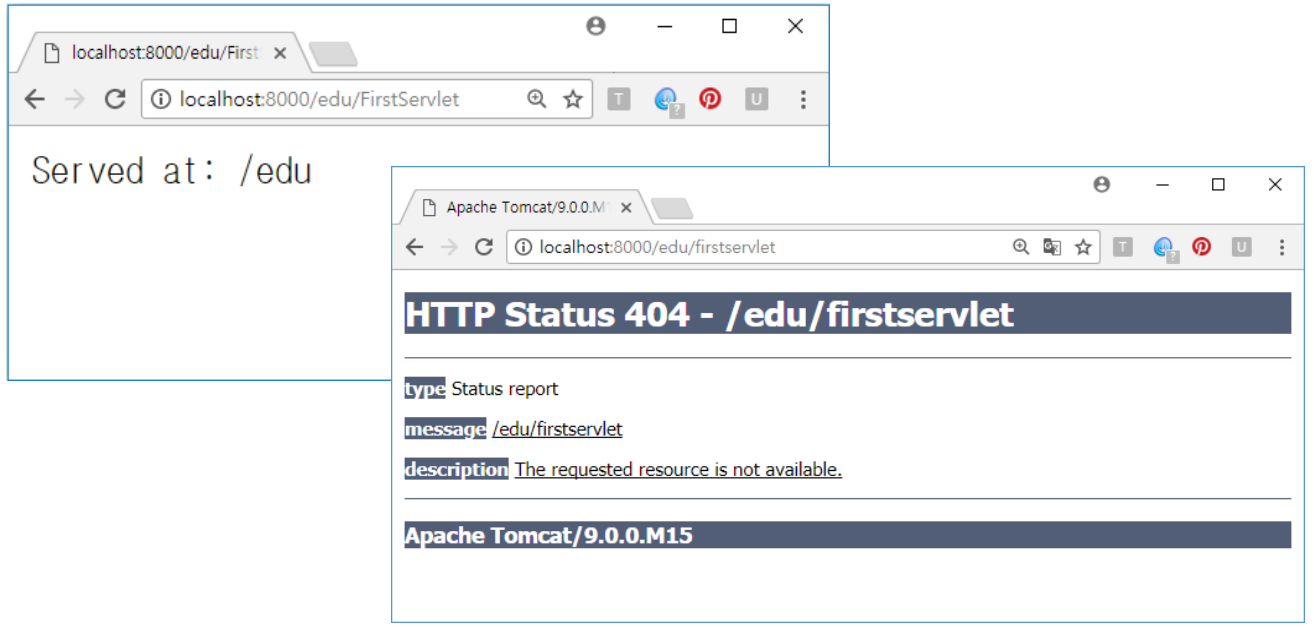
✓ Tomcat 서버가 정상적으로 기동되면 다음과 같이 Tomcat v9.0 Server at localhosts [Started, Synchronized]를 출력



✓ 주소 필드에 http://localhost:8000/edu/FirstServlet을 입력하여 FirstServlet의 수행을 요청

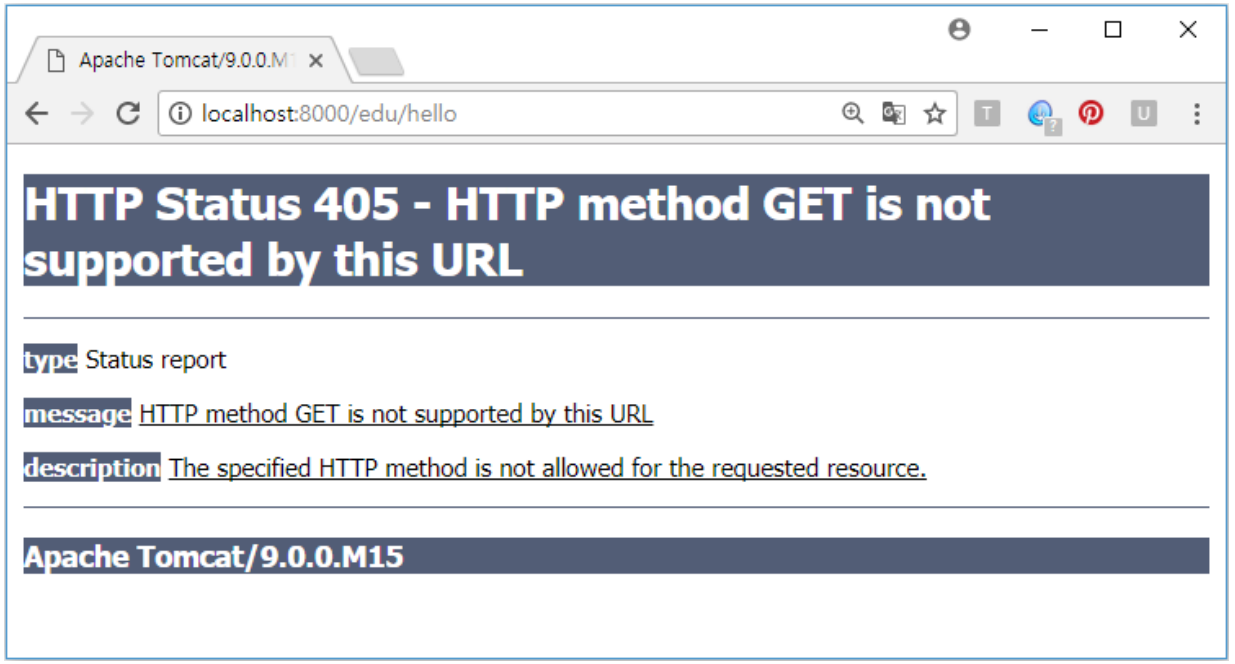
✓ HTTP Status 404 오류로 응답

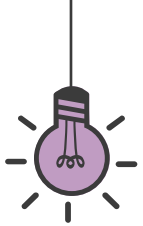
- (요청된 파일을 못 찾겠다는 것을 브라우저에게 알려주는 것)



1-1. Servlet의 구현

- ✓ 브라우저에서 URL을 주소 필드에 직접 입력하여 요청하는 것은 GET 방식 요청으로 간주
- ✓ doGet 메서드명을 doPost로 변경하여 테스트해보면 다음과 같이 405 응답 상태 코드 페이지를 응답





1-1. Servlet의 구현

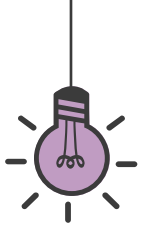
- ✓ Servlet 3.0부터는 web.xml에 작성하던 Servlet 등록과 매핑, 초기 파라미터 설정, 리스너나 필터 등록과 같은 내용들을 Servlet 소스 내에서 애노테이션 구문으로 구현 방법을 지원

애노테이션명	기능
@WebServlet	Servlet 프로그램의 등록과 매핑(URL mappings)을 정의함
@WebInitParam	Servlet 프로그램에 전달할 초기 파라미터를 정의함
@WebListener	리스너를 정의함
@WebFilter	필터를 정의함
@MultipartConfig	Servlet 프로그램에서 다중 파티션으로 전달되는 파일 업로드를 처리할 수 있음을 정의함

- ✓ 사용하는 애노테이션은 @WebServlet은 Servlet에 대한 URL mappings명을 설정하는 기능으로서 두 번째 예와 같이 URL mappings명을 2개 이상 설정하는 것도 가능

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet { ..... }

@WebServlet({"/hello1", "/hello2"})
public class HelloServlet extends HttpServlet { ..... }
```



1-2. Servlet의 수행

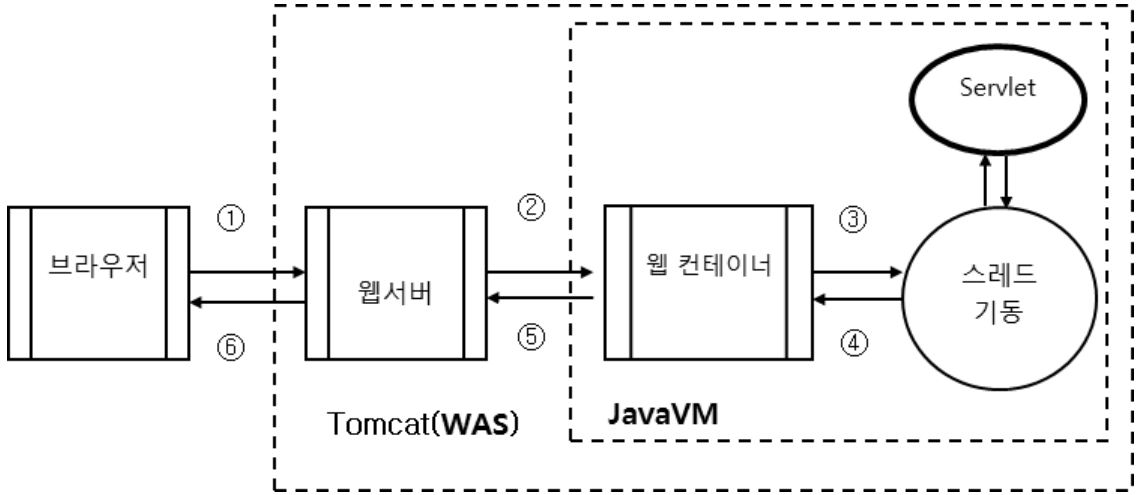
- ✓ 하나의 웹 컨테이너에 두 개 이상의 웹 애플리케이션이 배치(deploy)될 수 있으며 웹 애플리케이션 단위로 수행하고 관리
- ✓ 웹 컨테이너는 기동될 때 인식되는 웹 애플리케이션들을 하나 하나 컨텍스트 객체로 생성하여 관리
- ✓ 컨텍스트 객체마다 고유의 이름이 부여되는데 이를 컨텍스트명이라고 함
- ✓ 어떠한 웹 애플리케이션에서 제공되는 파일을 요청하는가에 따라서 요청 URL 작성 시 URI 맨 앞에 작성되는 패스가 바로 **"/" + 컨텍스트명**이 됨
- ✓ 교육용 Web 프로젝트로 edu라는 폴더를 생성하였고 Tomcat 서버에 등록
- ✓ Tomcat 서버는 edu 프로젝트를 edu라는 명칭의 컨텍스트로 인식하게 되므로 URL 작성 시 다음과 같이 구성하여 사용

http://localhost:8000/**edu**/XXXX

Eclipse	Dynamic Web Project 폴더
Tomcat	컨텍스트
개발자	웹 애플리케이션

1-2. Servlet의 수행

✓ 웹 클라이언트인 브라우저에서 Servlet 에 대한 수행이 요청되면 다음과 같은 순서대로 처리됨

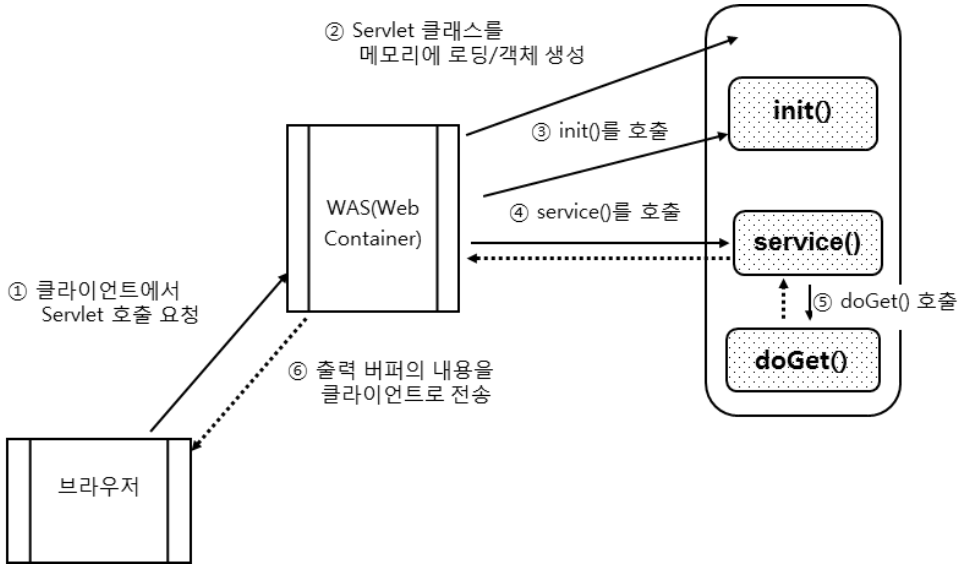


① 브라우저에서 Servlet을 요청하는 HTTP URL 문자열을 사용하여 수행을 요청함
② WAS 안의 웹서버가 Servlet 수행 요청임을 인식하고 웹 컨테이너에게 Servlet 수행을 요청함
③ 웹 컨테이너 스레드를 하나 준비하여 해당 Servlet을 수행함
④ 기동되었던 스레드를 반납하고 리턴함
⑤ Servlet의 수행 결과를 웹서버에 전송함
⑥ 웹서버는 Servlet의 수행 결과로 만들어진 출력 버퍼의 내용으로 HTTP 응답 헤더와 응답 바디를 구성하여 브라우저로 전송함

Servlet의 수행 흐름

1-2. Servlet의 수행

✓ Servlet이 컴파일된 후 첫 번째 요청일 때는 먼저 요청된 Servlet 클래스를 저장 장소에서 찾아 VM 영역에 로딩한 후 객체를 생성함 이때 생성되는 Servlet 객체는 웹 컨테이너가 종료될 때까지 또는 웹 애플리케이션이 리로드(Reload)될 때까지 메모리에 남아 있게 되며 다음과 같은 수행 흐름이 적용

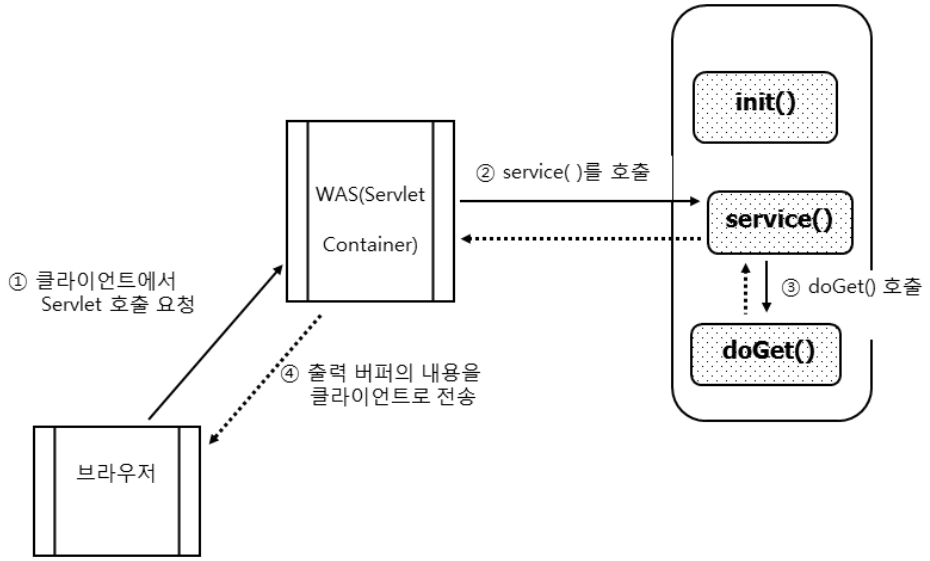


① 브라우저로부터 Servlet 수행 요청이 전송됨
② 요청된 Servlet 클래스를 찾아서 JVM 영역에 로딩한 후에 객체를 생성함
③ init(ServletConfig) 메서드를 호출함
④ service(ServletRequest, ServletResponse) 메서드를 호출함
⑤ 요청 방식에 따라서 doGet(HttpServletRequest, HttpServletResponse) 또는 doPost(HttpServletRequest, HttpServletResponse) 메서드를 호출함
⑥ 요청을 보낸 브라우저로 출력 버퍼의 내용을 리턴함

최초 요청 시 Servlet 수행 흐름

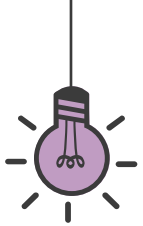
1-2. Servlet의 수행

✓ 두 번째 이후 요청부터는 이미 생성되어 있는 Servlet 객체를 찾아서 다음과 같은 순서를 적용하여 Servlet을 수행



① 브라우저로부터 Servlet 수행 요청이 전송됨
② service(ServletRequest, ServletResponse) 메서드를 호출함
③ 요청 방식에 따라서 doGet(HttpServletRequest, HttpServletResponse) 또는 doPost(HttpServletRequest, HttpServletResponse) 메서드를 호출함
④ 요청을 보낸 브라우저로 출력 버퍼의 내용을 리턴합니다

두 번째 이후 요청 시 Servlet 수행 흐름

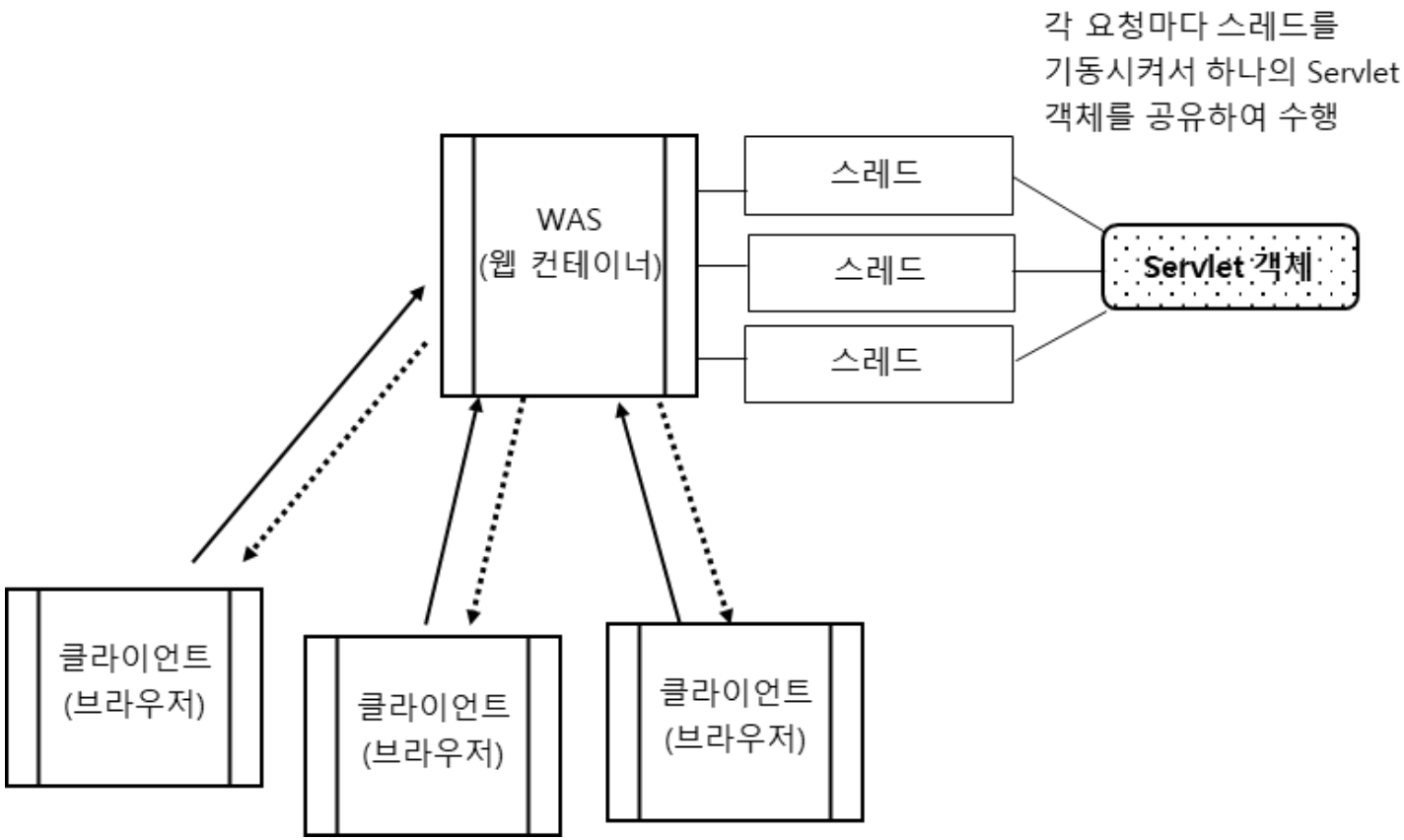


1-2. Servlet의 수행

- ✓ destroy( )는 브라우저로부터의 요청과 관계없이 메모리에 생성된 Servlet 객체가 메모리에서 해제되는 시점에 호출되는 메서드
- ✓ Servlet 객체가 메모리에서 해제되는 시점
  - ● 컨테이너(서버)가 종료될 때
  - ● 웹 애플리케이션이 리로드될 때
  - ● 자동 리로드가 설정된 상태에서 Servlet이 재컴파일되었을 때

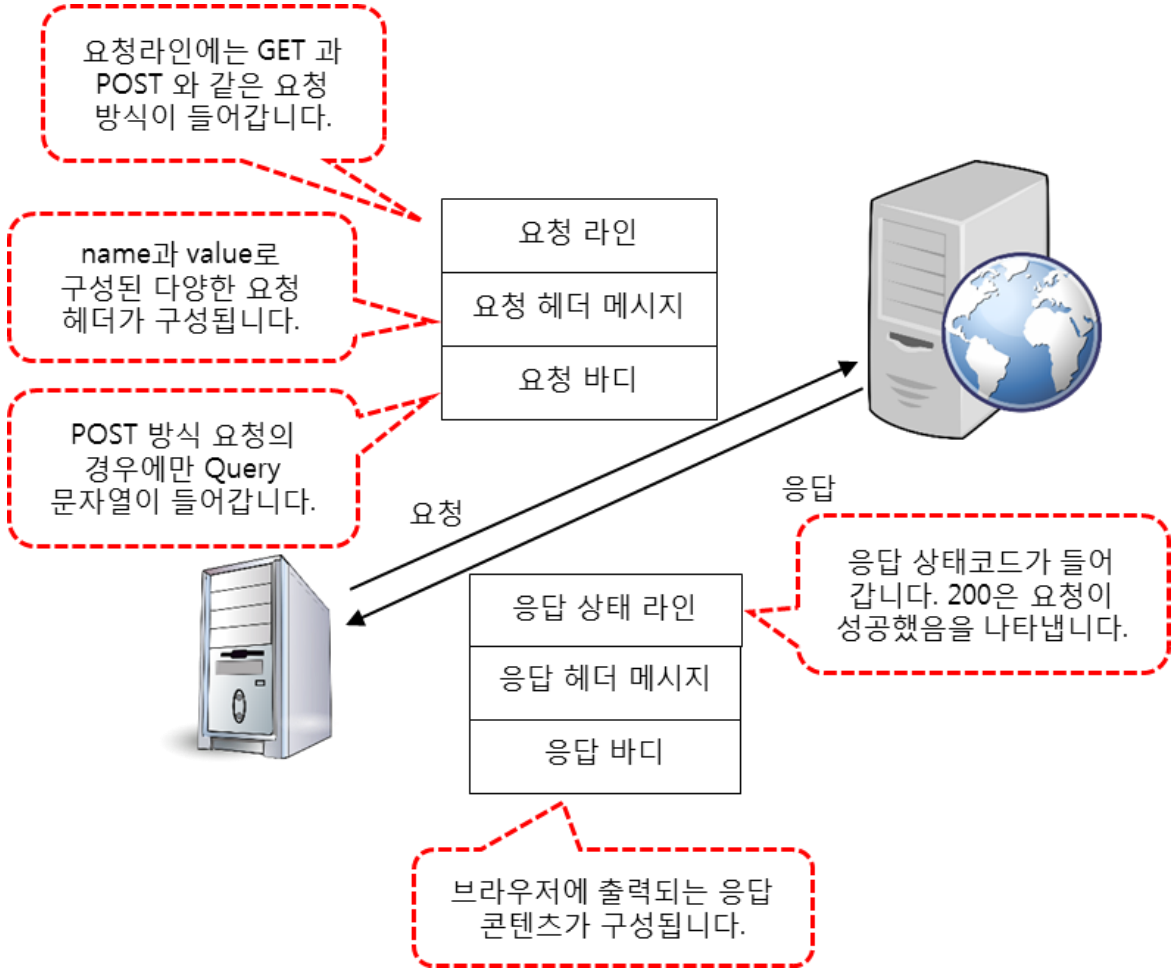
1-2. Servlet의 수행

- ✓ 웹 컨테이너가 Servlet 수행을 처리할 때 다중 스레드 방식을 사용
- ✓ 동일 Servlet을 여러 클라이언트가 동시에 요청하는 경우 각 요청마다 Servlet 클래스의 객체를 생성하여 수행하는 것이 아니라 하나의 Servlet 객체를 공유하여 여러 스레드가 병렬로 처리

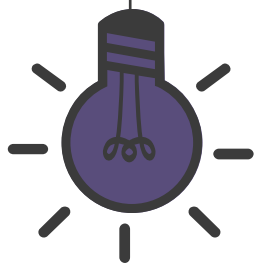


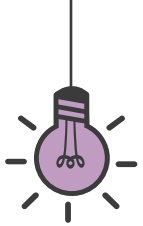
1-3. Servlet의 요청과 응답

- ✓ HTTP(HyperText Transfer Protocol)는 WWW(월드와이드웹)에서 정보를 주고받는 데 사용하는 통신 프로토콜(규약)
- ✓ HTTP는 클라이언트와 서버 사이에 이루어지는 요청/응답(request/response) 프로토콜로서 클라이언트인 웹브라우저의 사용자가 http:으로 시작하는 URL 문자열을 사용해서 서버에 요청을 보내면 서버가 요청에 대한 응답을 전달
- ✓ HTTP에서는 클라이언트에서 서버로 요청할 때 그리고 서버에서 클라이언트로 응답할 때 다음과 같은 형식의 요청 메시지와 응답 메시지를 구성



02 Query 문자열 처리





Query 문자열 처리

- ✓ Query 문자열이란 웹 클라이언트가 웹서버에게 요청을 보내면서 함께 전달하는 추가 문자열로서 name과 value로 구성되며 요청 파라미터라고도 함 우리가 웹에서 회원가입, 로그인 그리고 제품을 선택하여 장바구니에 담기 등의 행위는 모두 Query 문자열을 사용
- ✓ 다음과 같은 형식으로 전달되는데 전달 방식은 GET 방식과 POST 방식이 있음

```
name1=value1&name2=value2&name3=value3.....
```

2-1. GET 방식 Query 문자열 처리

- ✓ HTTP 최초 버전에서는 GET 방식만 지원
- ✓ "지정된 URI에 대한 콘텐츠를 달라"는 의미로서 다음과 같이 대부분의 요청은 GET 방식이 기본
  - ● 하이퍼링크 텍스트(<A> 태그)를 클릭하여 요청함
  - ● URL을 주소 필드에 입력하여 요청함
  - ● <IMG> 태그로 요청함
  - ● method 속성이 GET으로 설정된 <FORM> 태그로 요청함
  - ● <IFRAME> 태그로 요청함
- ✓ GET 방식 요청은 Query 문자열이 없는 요청과 Query 문자열을 추가한 요청 모두 가능
- ✓ Query 문자열을 추가하여 요청하는 경우 다음과 같은 특징이 있음
  - ● 전달되는 Query 문자열의 길이에 제한이 있고 내용이 브라우저의 주소 필드에 보임
  - ● <FORM> 태그를 사용해도 되고 요청 URL에 '?' 기호와 함께 직접 Query 문자열을 붙여서 전달하는 것도 가능함

2-1. GET 방식 Query 문자열 처리

- ✓ name과 value의 쌍으로 구성되는 Query 문자열 전체를 추출하려면 HttpServletRequest 객체의 getQueryString( )을 사용
- ✓ 대부분은 name에 value가 1개인 경우에는 주로 HttpServletRequest 객체의 getParameter(name)를 사용하고 동일한 name으로 여러 value 값이 전달되는 경우에는 HttpServletRequest 객체의 getParameterValues(name)를 사용

```
name으로 하나의 value 값이 전달될 때
                    String address = request.getParameter("address");
name으로 여러 개의 value 값들이 전달될 때
                    String hobby[ ] = request.getParameterValues("hobby");
```

2-2. POST 방식 Query 문자열 처리

- ✓ POST 요청 방식은 GET 방식을 보완하기 위해 추가된 요청 방식
- ✓ GET 방식과 다르게 전달되는 Query 문자열의 길이에 제한이 없고 내용이 브라우저의 주소 필드에 보이지 않음 POST 요청 방식은 Query 문자열이 요청 보디에 담겨 전달
- ✓ POST 방식의 요청은 <FORM> 태그를 사용하거나, JavaScript로 구현하여 요청할 수 있음
- ✓ Query 문자열의 추출은 GET 방식과 동일하게 request.getParameter(name) 또는 request.getParameterValues(name)를 사용
- ✓ GET 방식일 때와 다르게 POST 방식은 Query 문자열을 추출할 때 한글이 깨짐
- ✓ **Query 문자열을 추출하기 전에 HttpServletRequest에서 제공되는 setCharacterEncoding(“utf-8”) 을 호출**

2-3. multipart/form-data 처리(파일 업로드)

- ✓ HTML의 <form> 태그를 사용하여 전달되는 Query 문자열에는 사용자가 입력하거나 선택하는 내용과 함께 선택되는 파일도 추가하여 전달할 수 있음
- ✓ 클라이언트의 하드디스크에 존재하는 파일을 선택하여 서버에 전달하는 것을 파일 업로드라 함
- ✓ 파일 업로드를 처리하려면 우선 요청 방식은 POST 방식이어야 함
- ✓ <form> 태그의 서브 태그로 file 타입의 <input> 태그를 사용해서 파일을 선택할 수 있게 함
- ✓ <form> 태그에 method나 action 속성 외에 enctype이라는 속성을 사용해야 함
- ✓ enctype 속성은 서버에 보내지는 데이터의 형식을 지정하는 기능으로서 다음과 같이 세 가지의 종류 지원

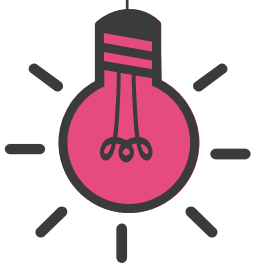
1. application/x-www-form-urlencoded
   디폴트 값으로 서버로 전송되기 전에 url encoding 방식으로 인코딩 된다는 뜻
   name1=value1&name2=value2... 형식을 의미함
2. mutipart/form-data
   사용자가 입력하거나 선택된 파일의 내용을 여러 개의 파트로 나누어 전송되는 방식으로서 파일 업로드 시에는 이 방식을 사용
3. text/plain
   인코딩을 하지 않은 문자 그대로의 상태를 전송한다는 의미

2-3. multipart/form-data 처리(파일 업로드)

- ✓ @MultipartConfig는 클라이언트에서 전송되는 multipart/form-data 형식의 데이터를 처리하는 Servlet이 정의해야 하는 애노테이션
- ✓ HttpServletRequest 객체의 getPart( ) 또는 getParts( ) 메서드를 사용하여 여러 개의 파트로 구성되어 전달되는 데이터를 간단하게 추출할 수 있음

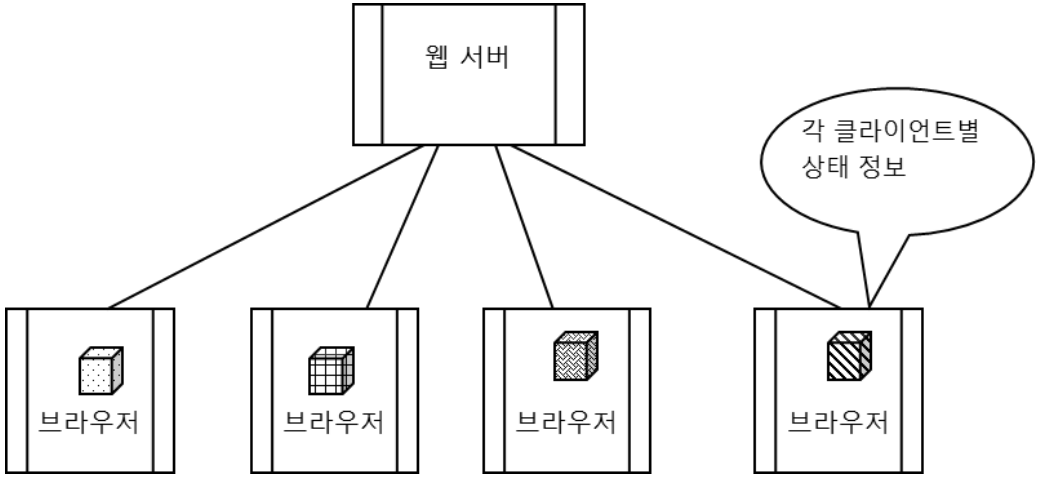
```
@MultipartConfig (location = "c:/uploadtest", maxFileSize = 1024 * 1024 * 5, maxRequestSize = 1024 * 1024 * 5 * 5)
```

03 상태 정보 관리

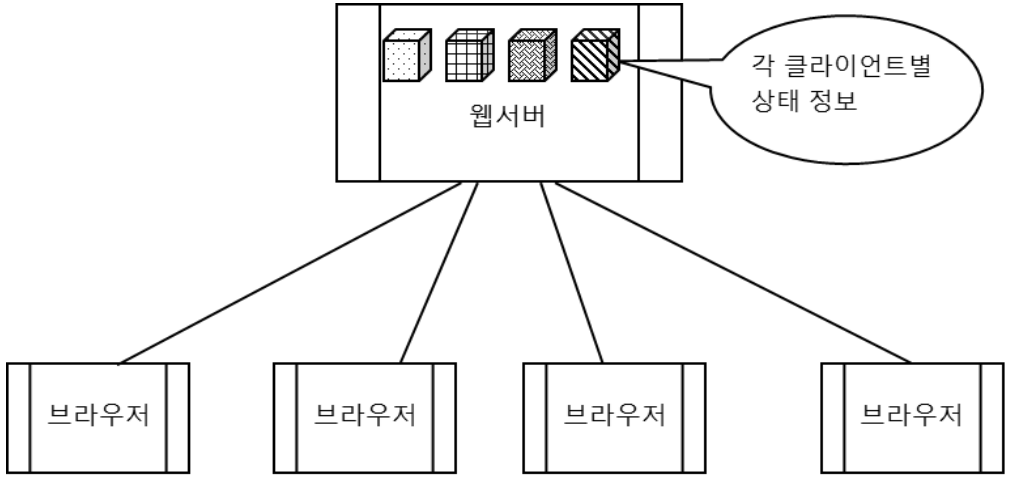


상태 정보 관리

- ✓ 웹의 통신 프로토콜은 요청-응답 프로토콜이라고 했음 웹브라우저에서 웹서버에 정보를 요청하면서 만들어진 결과물을 상태 정보라고 함
- ✓ HTTP 프로토콜은 기본적으로 Stateless 프로토콜로서 접속이 종료되면 이전 접속 시 생성된 모든 정보를 유지하지 않음
- ✓ 일정 시간 동안 이전 접속 시 결과물 즉, 상태 정보를 유지하는 것을 상태 정보 유지라고 함



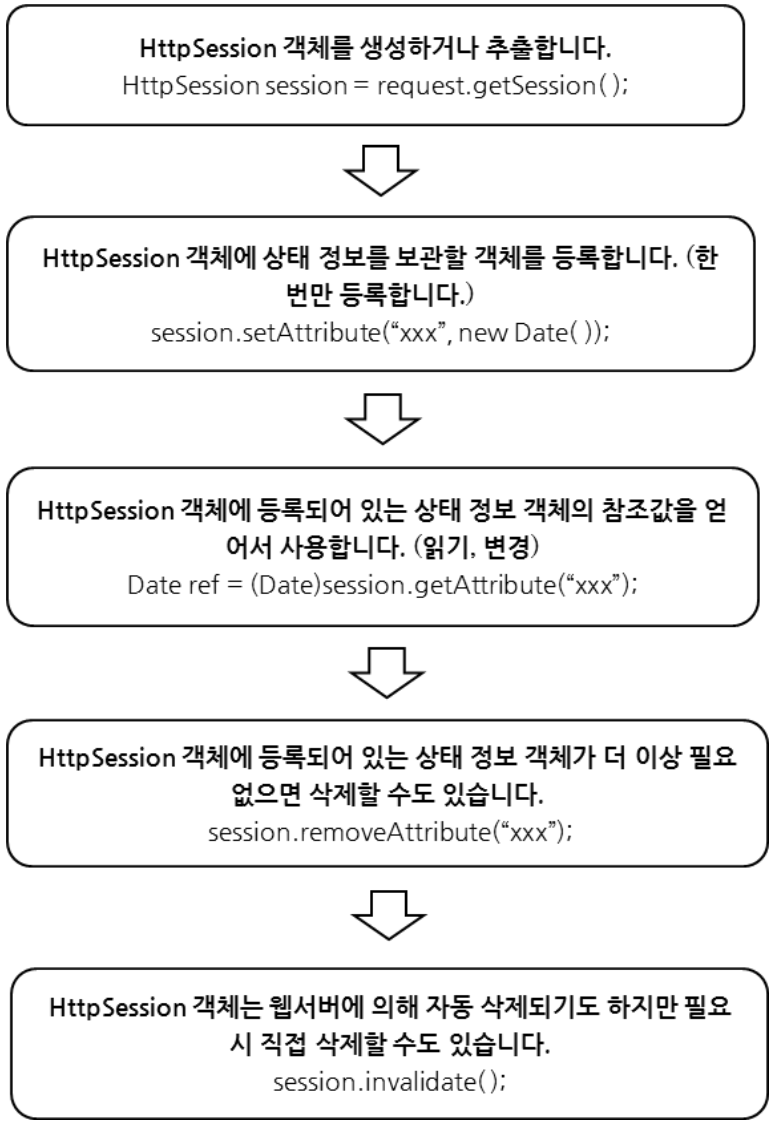
Cookie 기술: 클라이언트 저장



Cookie 기술: 클라이언트 저장

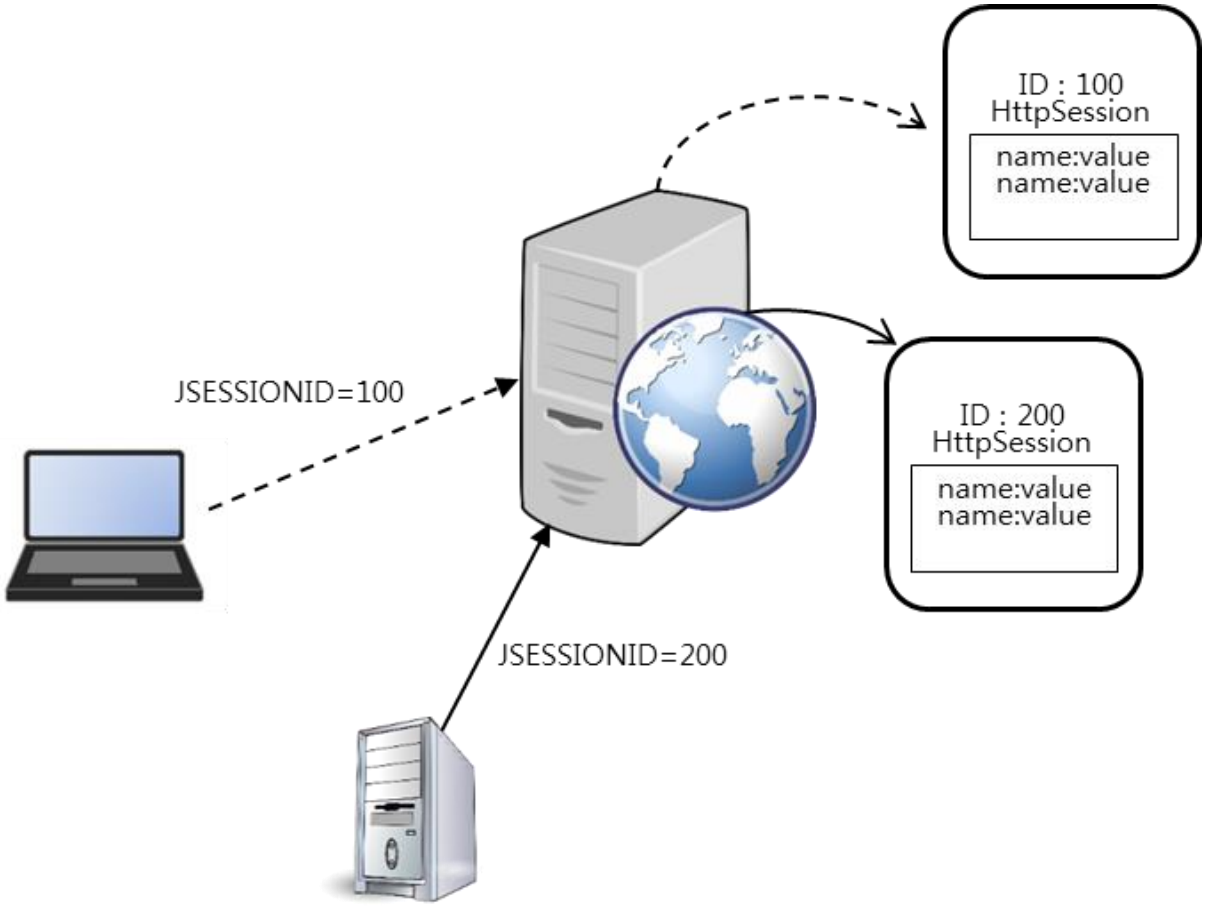
3-1. HttpSession 객체를 활용하는 상태 정보 관리

✓ HttpSession을 이용한 상태 정보 유지 구현 과정



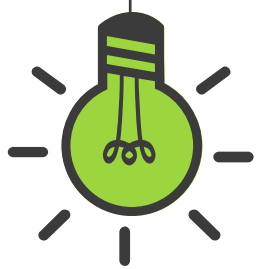
3-1. HttpSession 객체를 활용하는 상태 정보 관리

- ✓ HttpSession 객체가 생성될 때 세션 ID 가 하나 부여되며 이 세션 ID는 요청을 보내온 클라이언트의 브라우저에 Cookie 기술을 이용하여 JSESSIONID 라는 이름으로 저장되며 브라우저는 서버에 요청을 보낼 때마다 이 Cookie 정보를 요청 헤더에 담아 서버로 전송



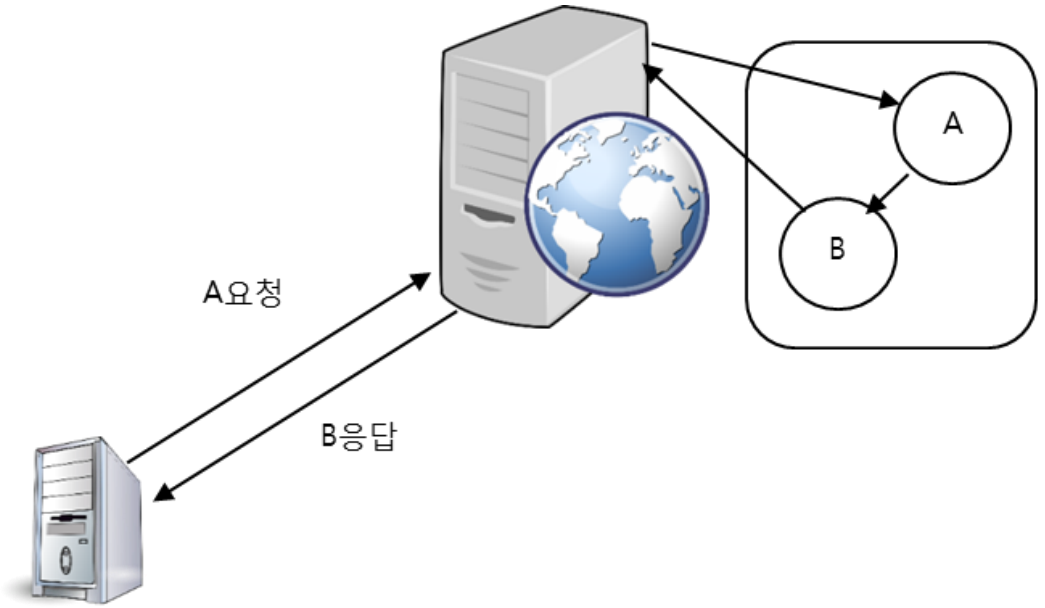
HttpSession 객체를 활용한 상태 정보 저장

04 요청 재지정



요청 재지정

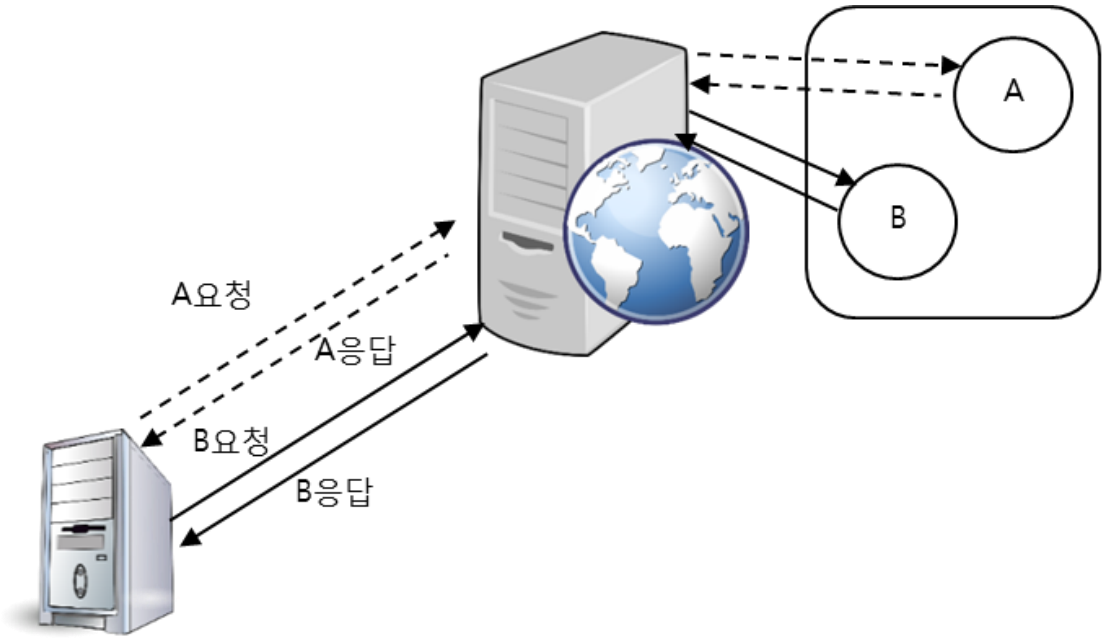
- ✓ 요청 재지정이란 클라이언트에서 요청한 Servlet의 응답 대신 다른 자원(Servlet, JSP, HTML 등)의 수행 결과를 클라이언트에 대신 응답하는 기능
- ✓ redirect 방법과 forward 방법으로 나뉨
- ✓ 클라이언트로부터 수행을 요청받은 A가 수행 권한을 B에게 넘겨서 대신 응답하게 함
- ✓ 클라이언트에서는 요청이 재지정된 사실을 알 수 없음
- ✓ forward는 동일 서버에서도 동일 웹 애플리케이션의 자원으로만 요청을 재지정할 수 있음



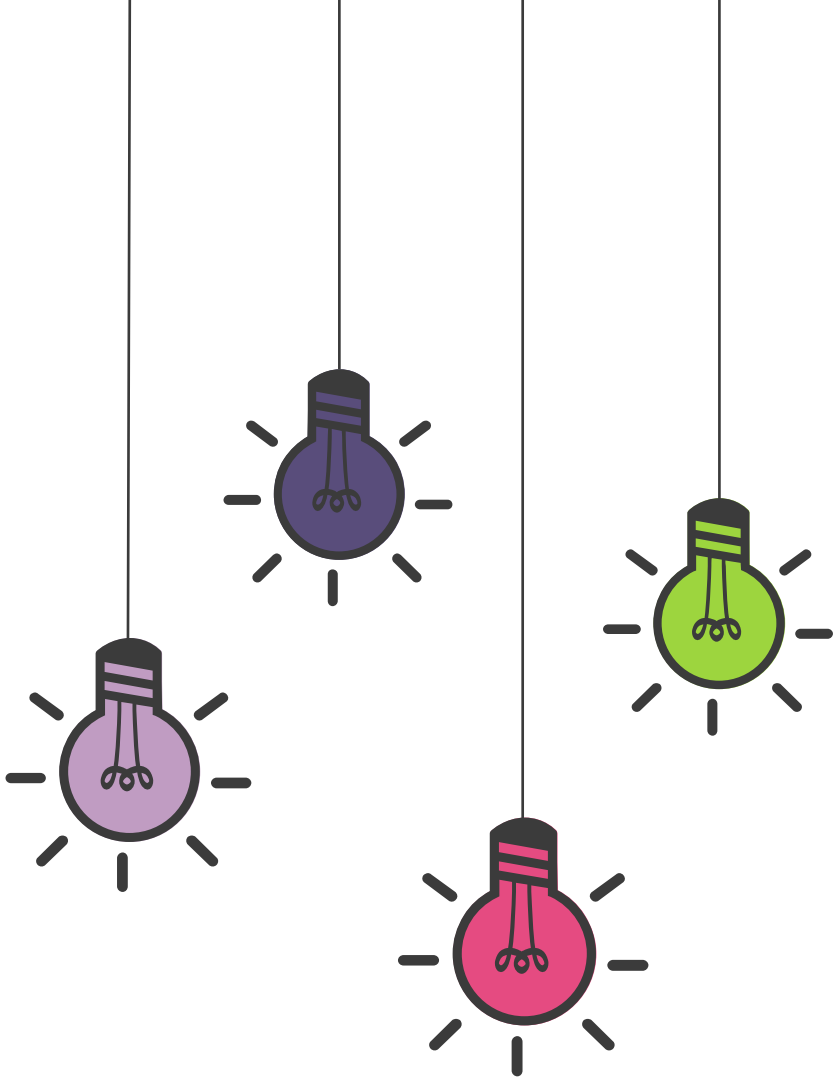
forward 방식의 요청 재지정

요청 재지정

- ✓ 다음 그림은 redirect 기능을 설명하는 그림
- ✓ 클라이언트로부터 수행을 요청받은 A가 302라는 응답코드와 재요청할 B자원에 대한 URL 정보를 가지고 응답
- ✓ 브라우저는 응답된 내용을 파악하여 서버에 B자원을 재요청함 302라는 응답 코드의 의미는 "요구한 데이터가 변경된 URL에 있음을 명시"라는 의미
- ✓ 브라우저가 직접 요청하여 응답받게 되므로 브라우저 주소 필드의 URL 문자열이 B에 대한 내용으로 변경
- ✓ 클라이언트 사용자는 요청이 재지정된 사실을 알게 됨
- ✓ redirect는 동일 서버뿐만 아니라 다른 웹사이트의 자원으로도 요청을 재지정할 수 있음



redirect 방식의 요청 재지정



감사합니다

THANK YOU FOR WATCHING