KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

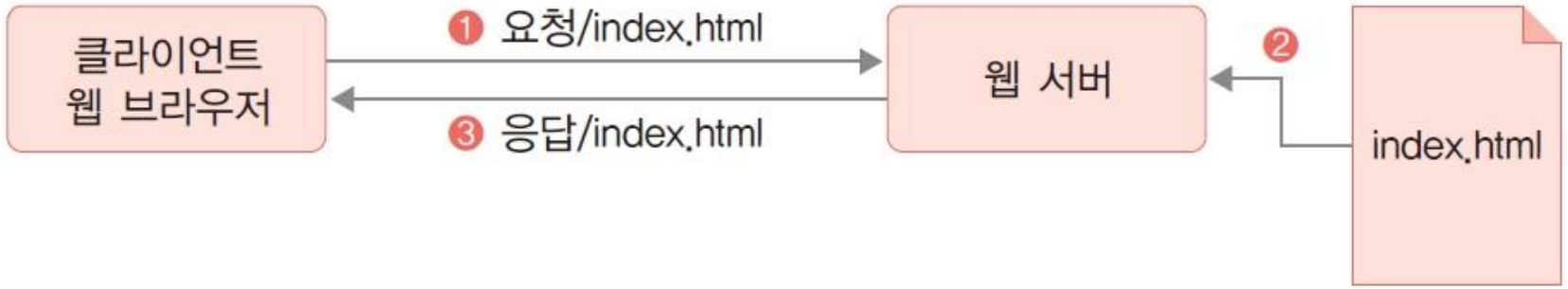
서블릿의 이해

[KB] IT's Your Life

KB 국민은행

서블릿(Servlet) 개요

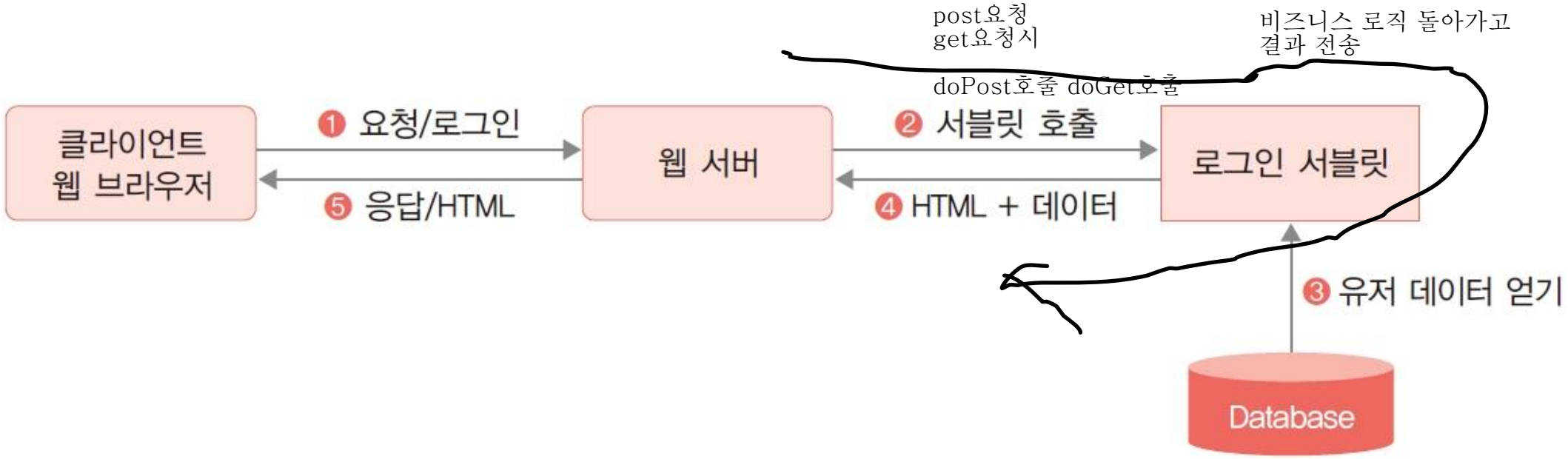
기본적인 웹의 요청과 응답 과정



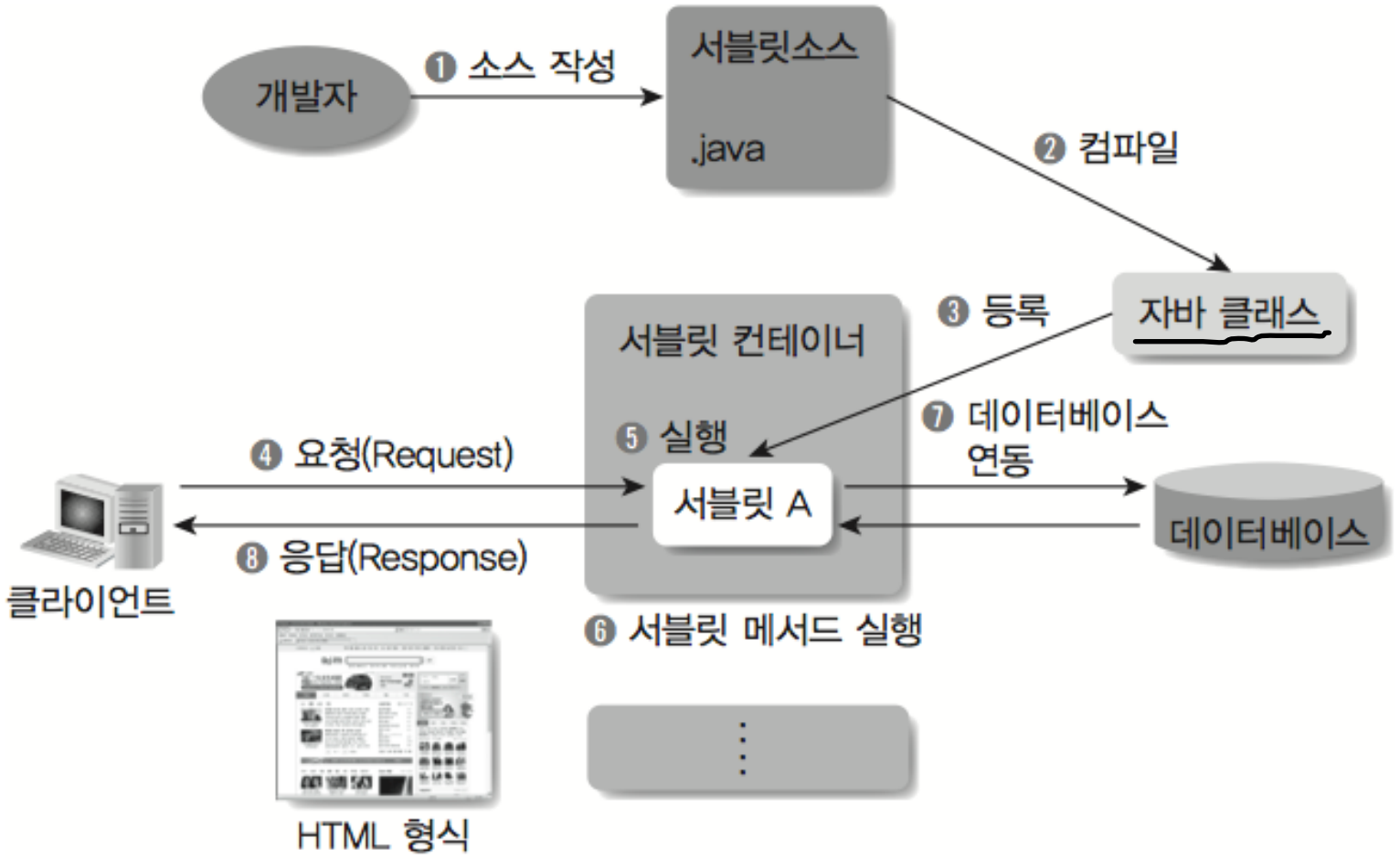
서블릿(Servlet) 개요

서블릿

- 웹 컨테이너에 의해 관리
- 다양한 클라이언트 요청에 의해 컨텐츠로 응답 가능한 자바 기반의 웹 컴포넌트



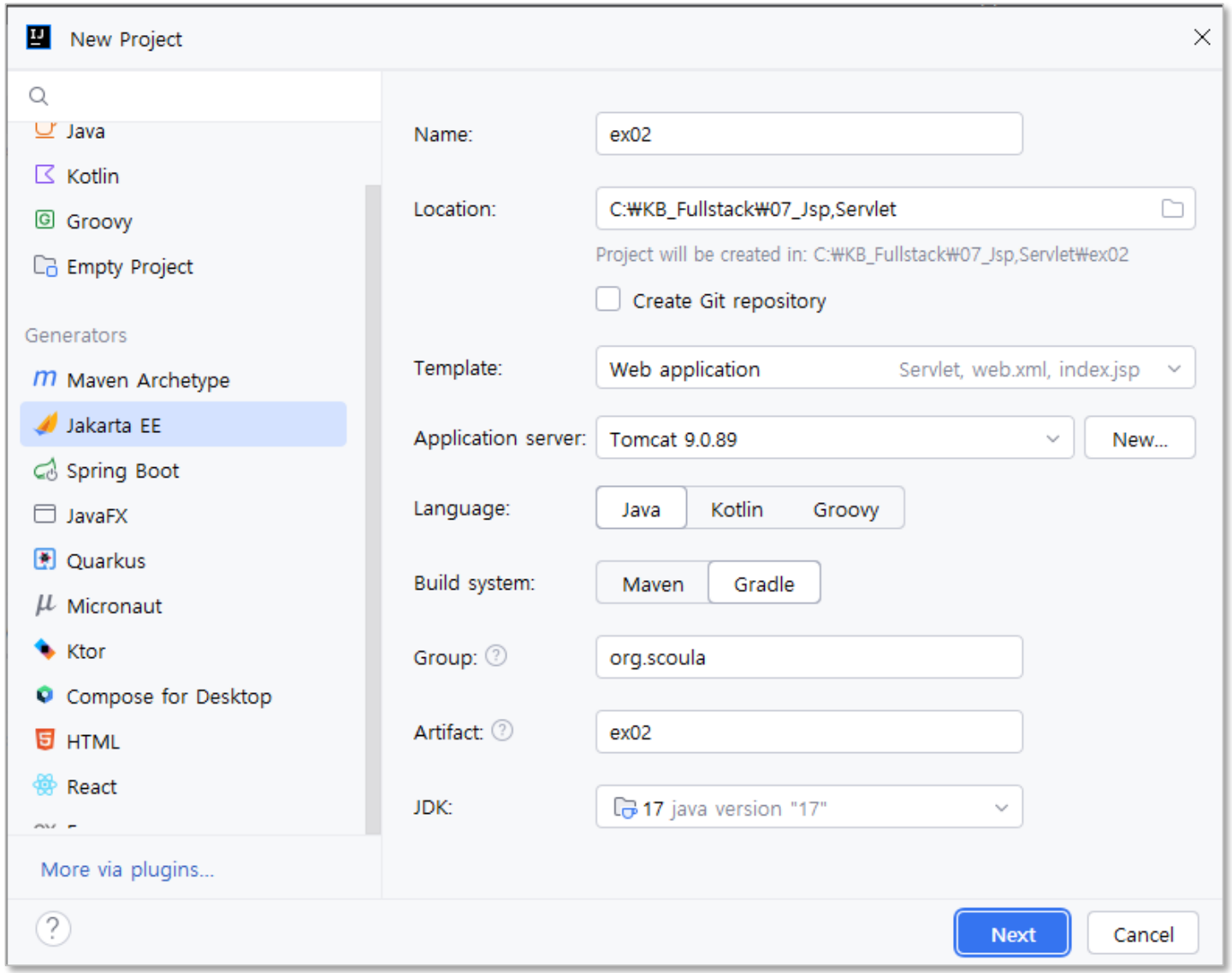
서블릿(Servlet) 개요



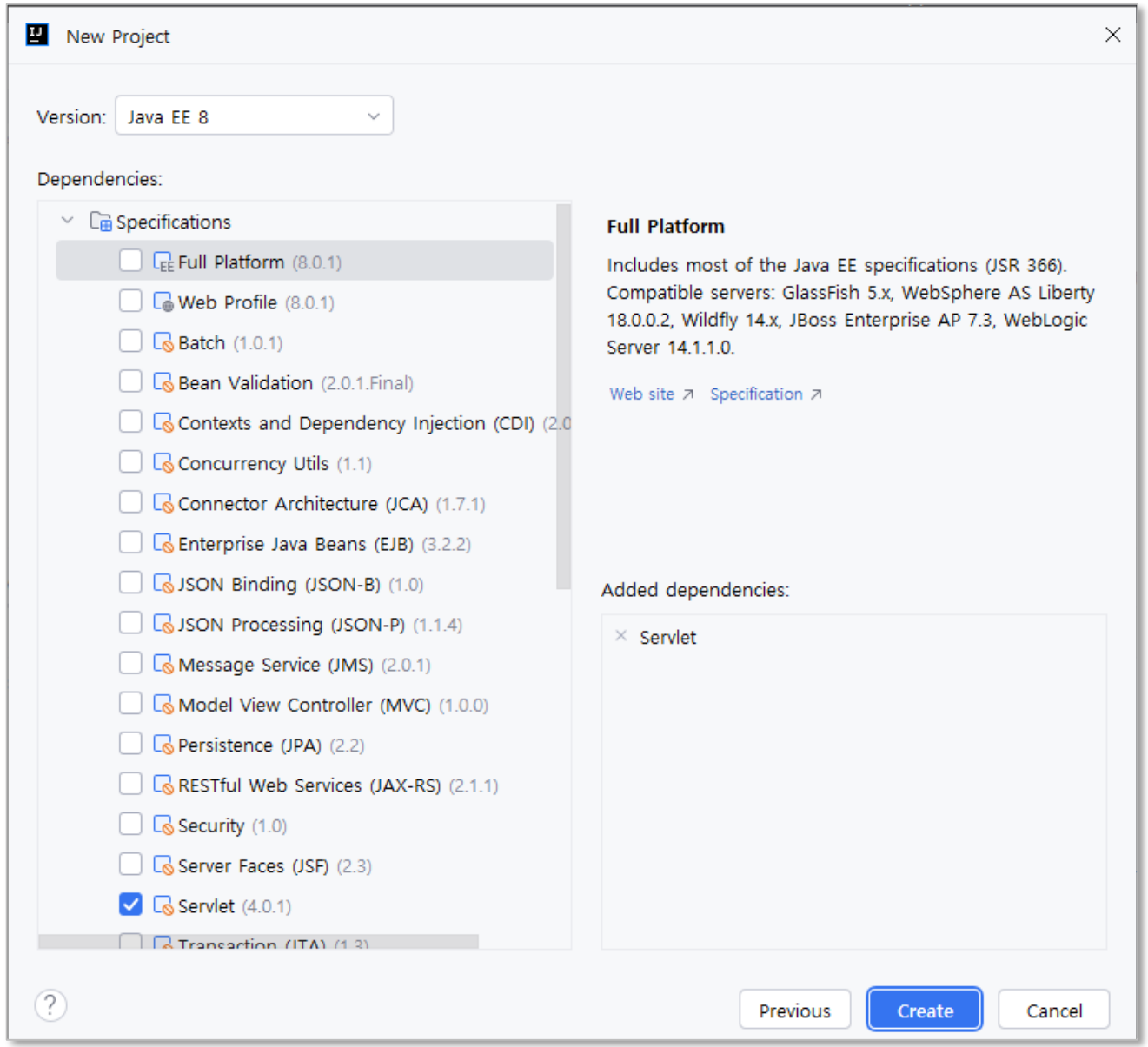
프로젝트 만들기

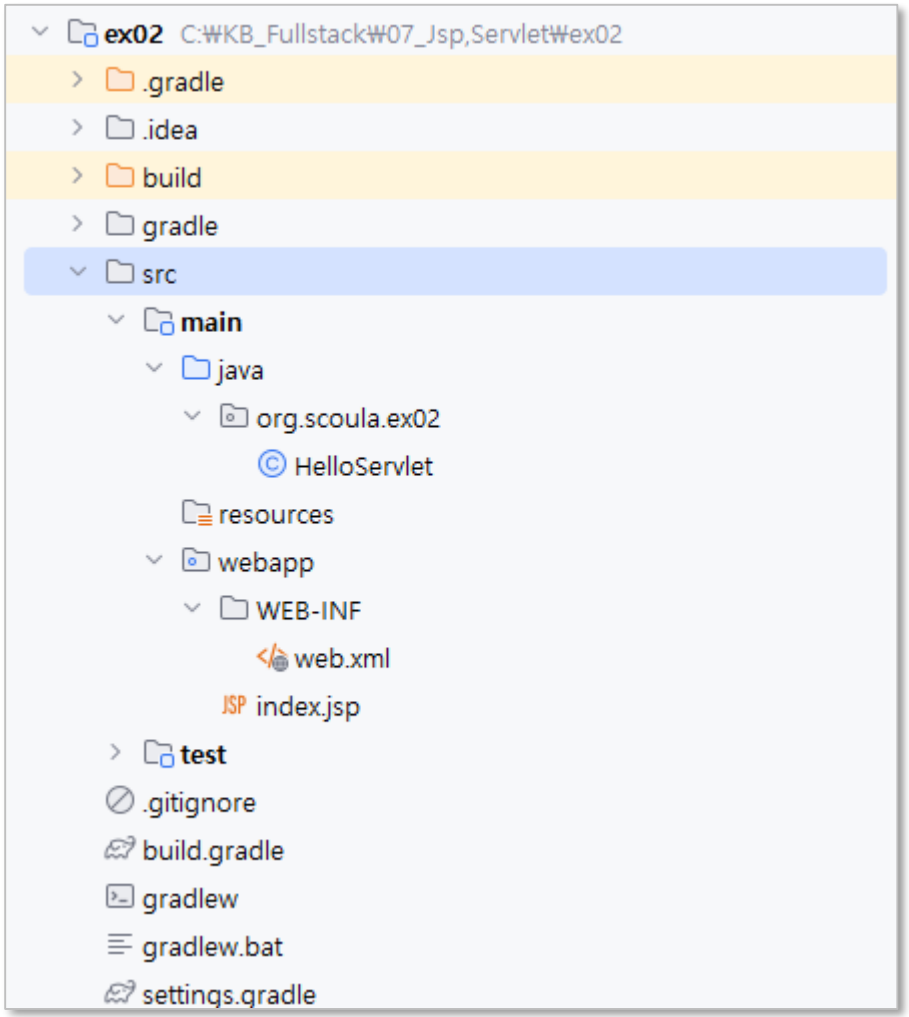
프로젝트 생성

- Templates: Jakarta EE
- Project name: ex02



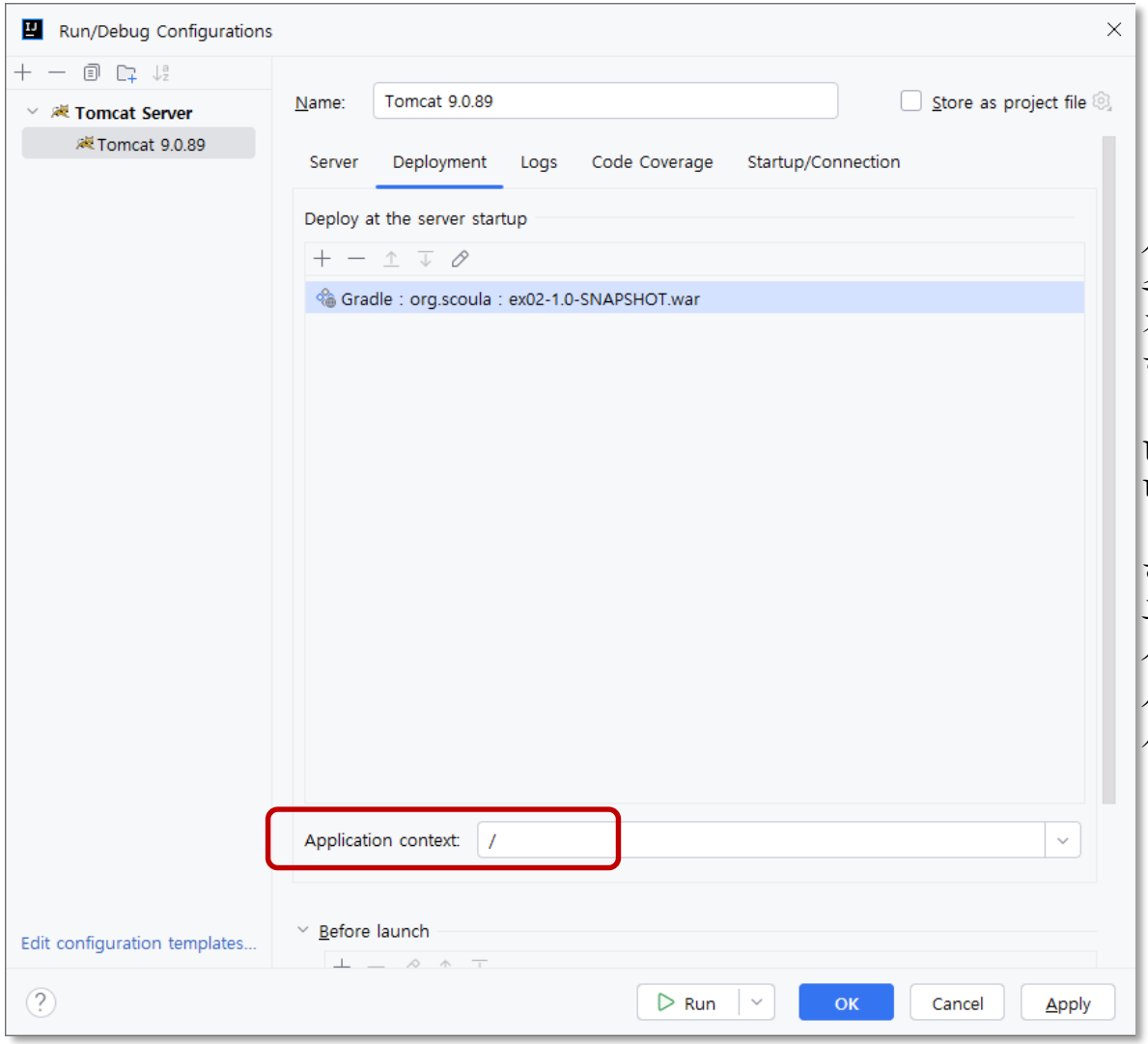
프로젝트 만들기


New Project
Version: Java EE 8
Dependencies:
Specifications
Full Platform (8.0.1)
Web Profile (8.0.1)
Batch (1.0.1)
Bean Validation (2.0.1.Final)
Contexts and Dependency Injection (CDI)
Concurrency Utils (1.1)
Connector Architecture (JCA) (1.7.1)
Enterprise Java Beans (EJB) (3.2.2)
JSON Binding (JSON-B) (1.0)
JSON Processing (JSON-P) (1.1.4)
Message Service (JMS) (2.0.1)
Model View Controller (MVC) (1.0.0)
Persistence (JPA) (2.2)
RESTful Web Services (JAX-RS) (2.1.1)
Security (1.0)
Server Faces (JSF) (2.3)
Servlet (4.0.1)
Full Platform
Includes most of the Java EE specifications (JSR 366). Compatible servers: GlassFish 5.x, WebSphere AS Liberty 18.0.0.2, Wildfly 14.x, JBoss Enterprise AP 7.3, WebLogic Server 14.1.1.0.
Web site
Specification
Added dependencies:
Servlet
Previous
Create
Cancel


ex02 C:₩KB_Fullstack₩07_Jsp,Servlet₩ex02
.gradle
.idea
build
gradle
src
main
java
org.scoula.ex02
HelloServlet
resources
webapp
WEB-INF
web.xml
index.jsp
test
.gitignore
build.gradle
gradlew
gradlew.bat
settings.gradle

프로젝트 만들기

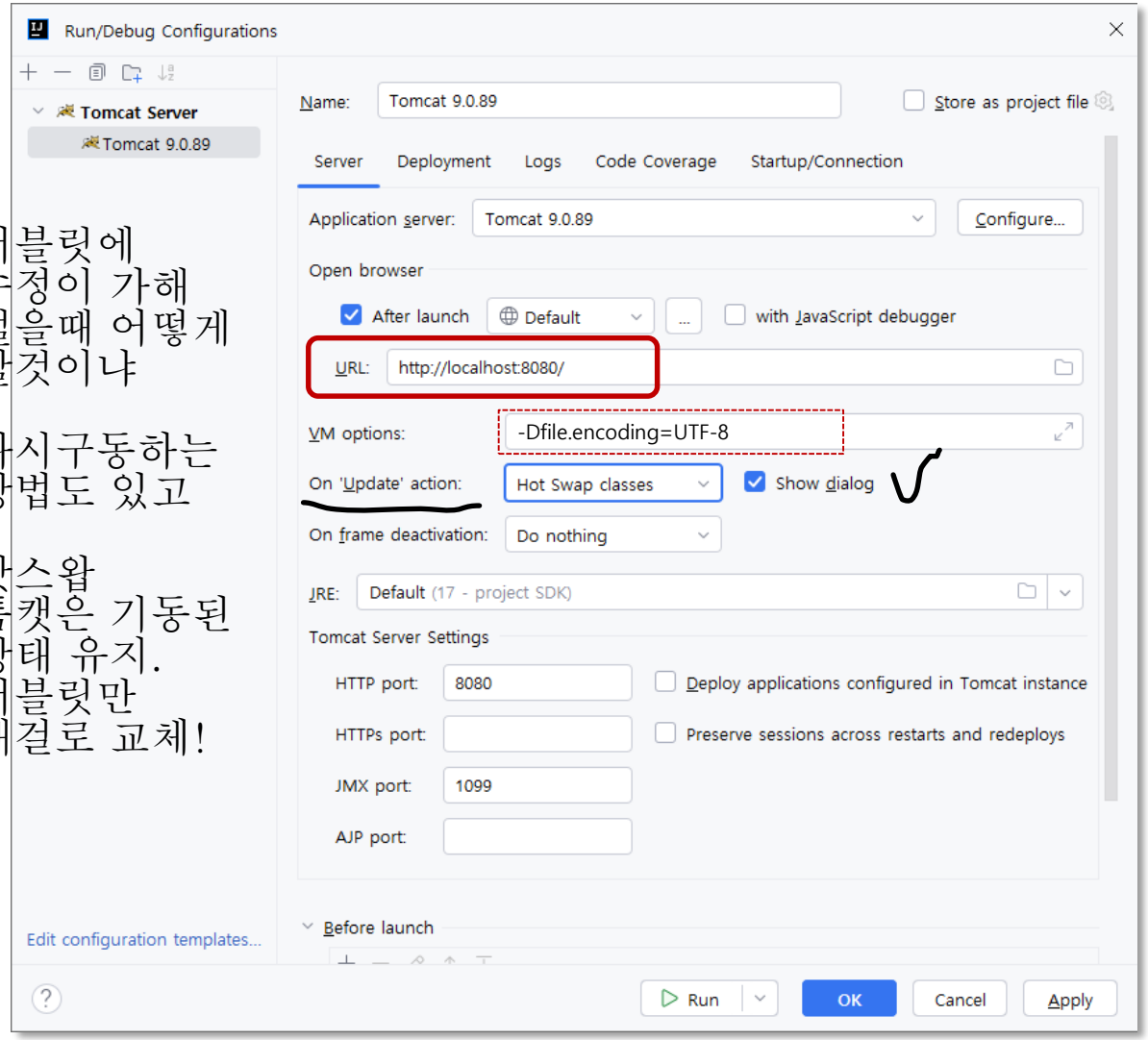
Tomcat 설정



서블릿에
수정이 가해
졌을때 어떻게
할것이냐

다시구동하는
방법도 있고

핫스왑
톰캣은 기동된
상태 유지.
서블릿만
새걸로 교체!

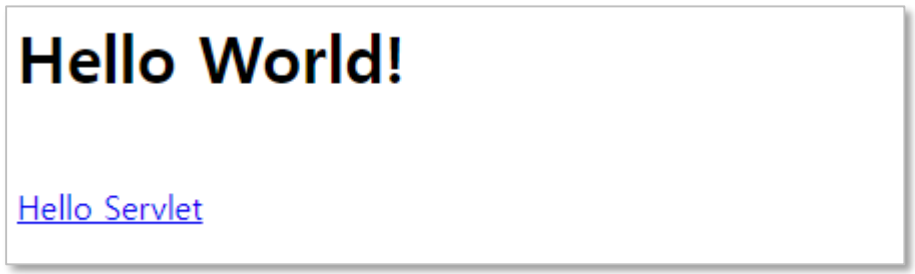


프로젝트 만들기

index.jsp

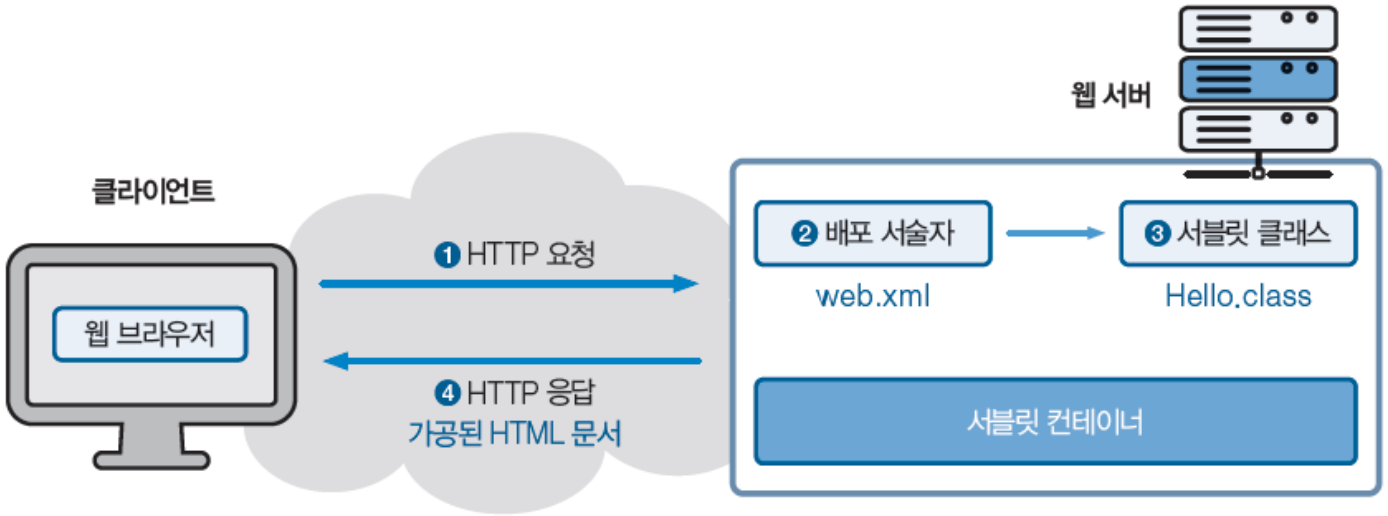
```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <title>JSP - Hello World</title>
</head>
<body>
<h1><%= "Hello World!" %>
</h1>
<br/>
<a href="hello-servlet">Hello Servlet</a>
</body>
</html>
```

http://localhost:8080/



서블릿 맵핑

서블릿 동작 과정



HelloServlet.java

```
package org.scoula.ex02;

import java.io.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;

// @WebServlet(name = "helloServlet", value = "/hello-servlet")
public class HelloServlet extends HttpServlet {
    private String message;

    public void init() {
        message = "Hello World!";
    }

    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {
        resp.setContentType("text/html");

        PrintWriter out = resp.getWriter();
        out.println("<html><body>");
        out.println("<h1>" + message + "</h1>");
        out.println("</body></html>");
    }

    public void destroy() {
    }
}
```

서블릿 맵핑

web.xml

서술자 기술자를 함보자잉

WA의 구성정보를 명시

```
〈servlet〉
    〈servlet-name〉서블릿별명〈/servlet-name〉
    〈servlet-class〉패키지를 포함한 서블릿명〈/servlet-class〉
〈/servlet〉
〈servlet-mapping〉
    〈servlet-name〉서블릿별명〈/servlet-name〉
    〈url-pattern〉/맵핑명〈/url-pattern〉
〈/servlet-mapping〉
```

```
package org.scoula.ex02;

…

// @WebServlet(name = "helloServlet", value = "/hello-servlet")
public class HelloServlet extends HttpServlet {
    …
}
```

서블릿 맵핑

web.xml

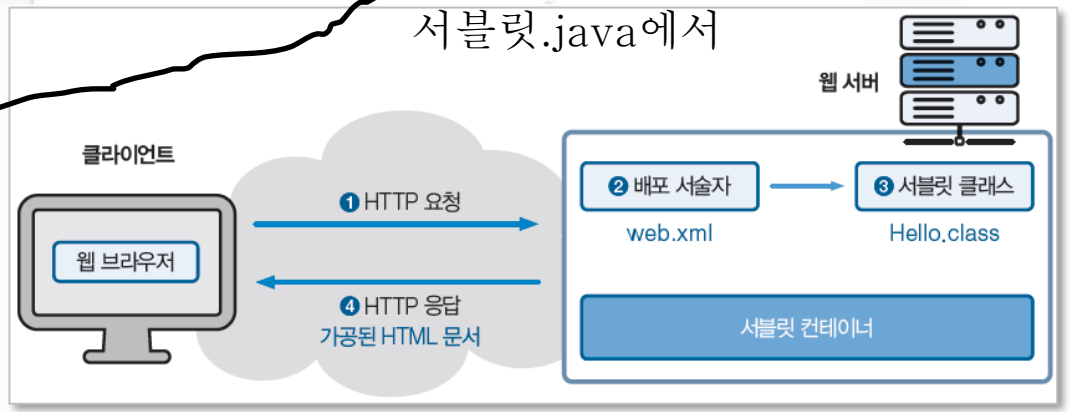
- web.xml 수정본을 반영하려면 웹서버 재기동

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         id="WebApp_ID" version="3.1">

    <servlet>
        <servlet-name>helloServlet</servlet-name>
        <servlet-class>org.scoula.ex02.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>helloServlet</servlet-name>
        <url-pattern>/hello-servlet</url-pattern>
    </servlet-mapping>

</web-app>
```

```
@WebServlet(name = "helloServlet", value = "/hello-servlet")
```

서블릿.java에서

거의다 코드가
어노테이션 기반으로 바뀜
그래서 web.xml이 할일이 없어짐.
그래서 요즘에 해당 파일도 생략 가능할수도
하지만 디폴트로는 빈 web.xml을 제공하긴함



너무 불편함.
매번 하기가.
실수도 잦고
그래서 어노테이션으로 바뀜

http://localhost:8080/

Hello World!

Hello Servlet

클릭시 /hello-servlet 의 이름인 서블릿을 찾아서 구동함

http://localhost:8080/**hello-servlet**

Hello World!

@WebServlet

@WebServlet 어노테이션 이용 방법

- Xml 파일 대신 자바 코드에 기술
- Servlet 3.0의 어노테이션 ✓

- @WebServlet ✓
- @WebFilter
- @WebInitParam
- @WebListener
- @MultipartConfig
- @DeclareRoles
- @EJB
- @EJBs
- @Resource
- @Resources
- @PersistenceContext
- @PersistenceContexts
- @PersistenceUnit
- @PersistenceUnits
- @PostConstruct
- @PreDestroy
- @RunAs

@WebServlet

서블릿 맵핑 설정

```
@WebServlet("/맵핑명")
public class MyServlet extends HttpServlet { … }
```

서블릿이름 생략시 클래스이름을 가지고 지정됨

서블릿 별명과 urlPatterns 속성으로 여러 개의 맵핑명 지정

```
@WebServlet(name="서블릿별명",
            urlPatterns={"/맵핑명", "/맵핑명2"})
public class MyServlet extends HttpServlet { … }
```

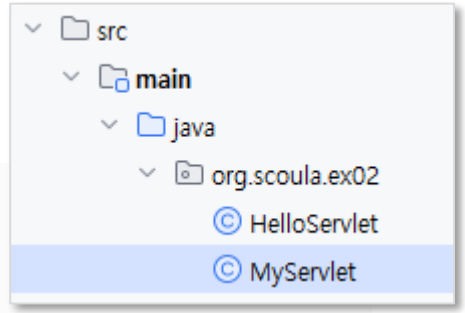
번외 어노테이션에서 중괄호표기는 배열 인스턴스을 나타냄

또는

```
@WebServlet(name="서블릿별명",
            value={"/맵핑명", "/맵핑명2"})
public class MyServlet extends HttpServlet { … }
```

@WebServlet

MyServlet.java 어노테이션을 이용한 서블릿 맵핑 등록




```
@WebServlet(name="MyServlet", urlPatterns={"/xxx", "/yyy" })
public class MyServlet extends HttpServlet {

  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {

    System.out.println("HelloServlet 요청");
    PrintWriter out = resp.getWriter();
    out.println("<h1>Hello Servlet</h1>");

  }
}
```

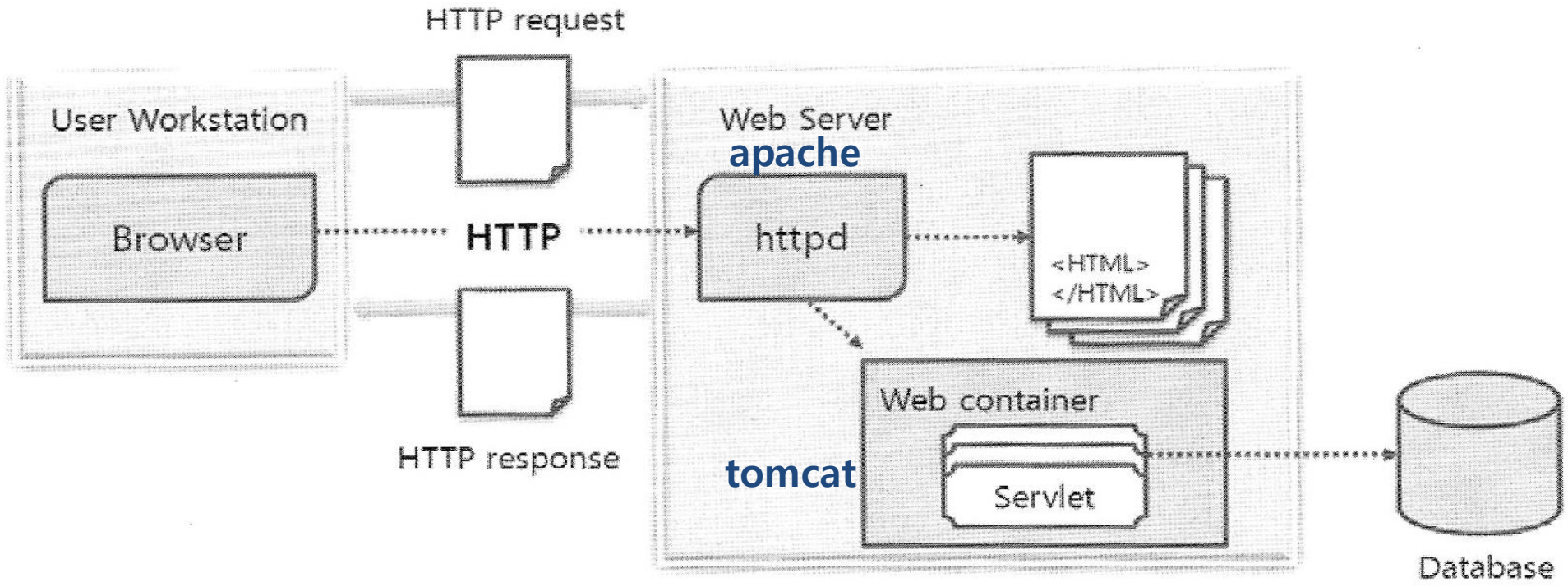
두요청 모두다 myservelet이 담당한다는 뜻

- http://localhost/xxx
- http://localhost/yyy


```
서버에 연결됨
[2025-01-10 03:39:23,532] 아티팩트 Gradle : org.scoula : ex02
[2025-01-10 03:39:24,082] 아티팩트 Gradle : org.scoula : ex02
[2025-01-10 03:39:24,082] 아티팩트 Gradle : org.scoula : ex02
10-Jan-2025 15:39:33.271 INFO [Catalina-utility-2] org.apac
10-Jan-2025 15:39:33.350 INFO [Catalina-utility-2] org.apac
HelloServlet 요청
```

서블릿 아키텍처 및 핵심 API

서블릿을 사용하는 경우 웹 아키텍처



서블릿 아키텍처 및 핵심 API

웹브라우저의 URL 요청 → 해당 서블릿 호출

- 웹 컨테이너에서 서블릿을 실행
- 결과값을 HTML로 구성하여 클라이언트에 응답

- HTTP Request(요청)
  - javax.servlet.http.HttpServletRequest
- HTTP Response(응답)
  - javax.servlet.http.HttpServletResponse

```
@WebServlet(name="MyServlet", urlPatterns={"/xxx", "/yyy" })
public class MyServlet extends HttpServlet {

  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {
     …
  }
}
```

서블릿 아키텍처 및 핵심 API

HttpServletRequest API

리퀘스트의 메서드를 보자잉

- getHeader(String name):String
- getHeaderNames( ):Enumeration
- getCookies( ):Cookie[ ]
- getRequestURI( ):String
- getServletPath( ):String
- getSession(boolean):HttpSession
- getSession( ):HttpSession
- setCharacterEncoding(String encoding)
- setAttriute(String name, Object obj)
- getAttribute(String name):Object
- removeAttribute(String name)
- getParameter(String name)
- getParameterNames( ):Enumeration
- getParameterValues(String name):String[ ]

서블릿 아키텍처 및 핵심 API

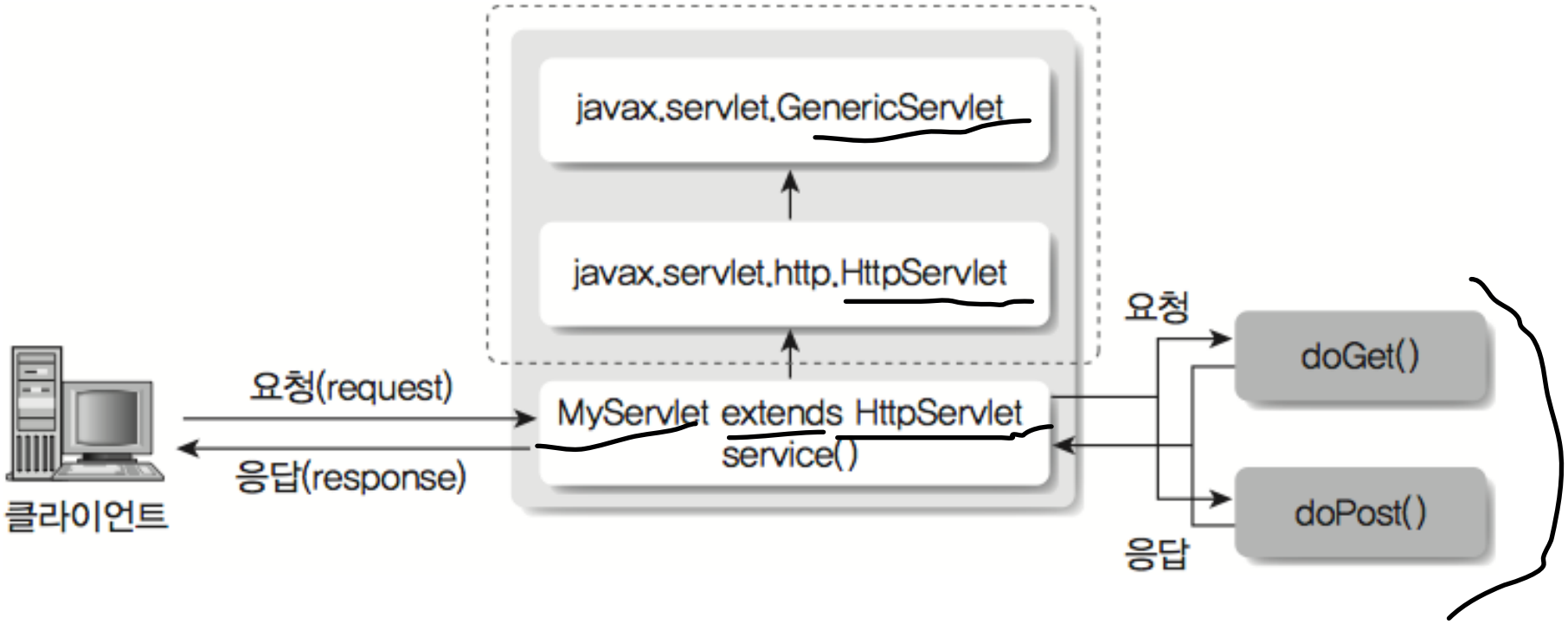
HttpServletResponse

응답 객체 메서드

- addCookie(Cookie c)
- addHeader(String name, String value)
- encodeURL(String url)
- getStatus( )
- sendRedirect(String loc)
- getWriter( ):PrintWriter
- getOutputStream( ):ServletOutputStream
- setContentType(String type)

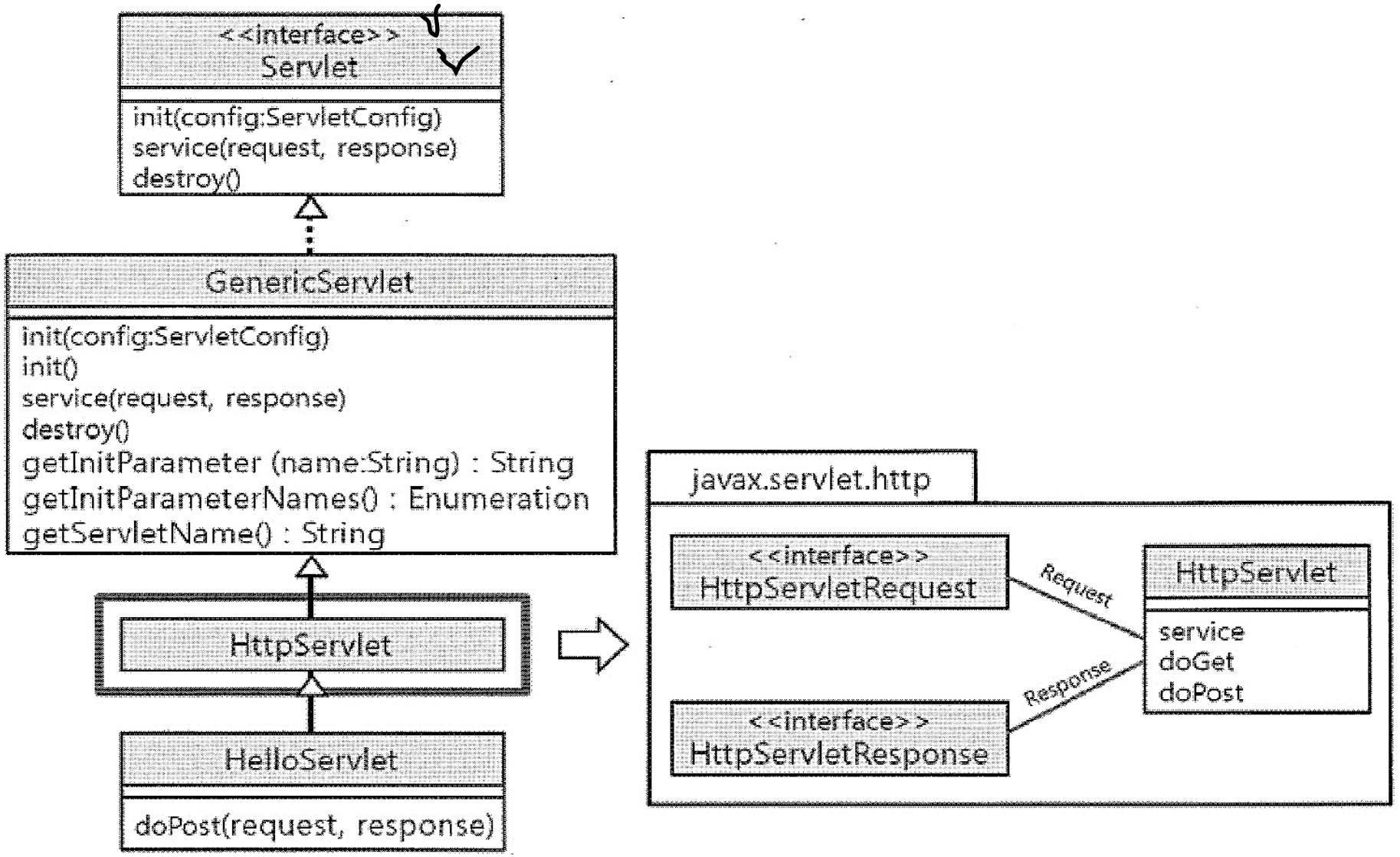
서블릿 응답 처리

- javax.servlet.http.HttpServlet을 상속받은 서블릿 동작 구조



서블릿 아키텍처 및 핵심 API

HttpServlet API



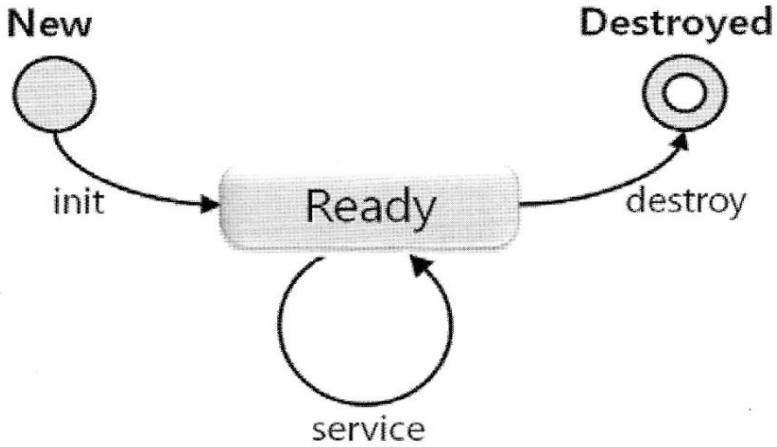
서블릿 LifeCycle 메서드

서블릿의 인스턴스를 init, service, destroy 메서드로 관리

기동되었다== 컨테이너에 등록될때
톰캣이 기동될때 호출됨. hot swap기동될때 호출됨

- **init 메서드**
  - 인스턴스가 처음 실행될 때, 단 한번 호출
  - 서블릿에서 필요한 초기화 작업 시 사용
- **service 메서드**
  - 클라이언트가 요청할 때마다 호출
  - doGet 또는 doPost에서 주로 작업
- **destroy 메서드**
  - 인스턴스가 웹 컨테이너에서 제거될 때 호출

주로 톰캣이 종료될때
혹은 컨테이너 제거 명시할때

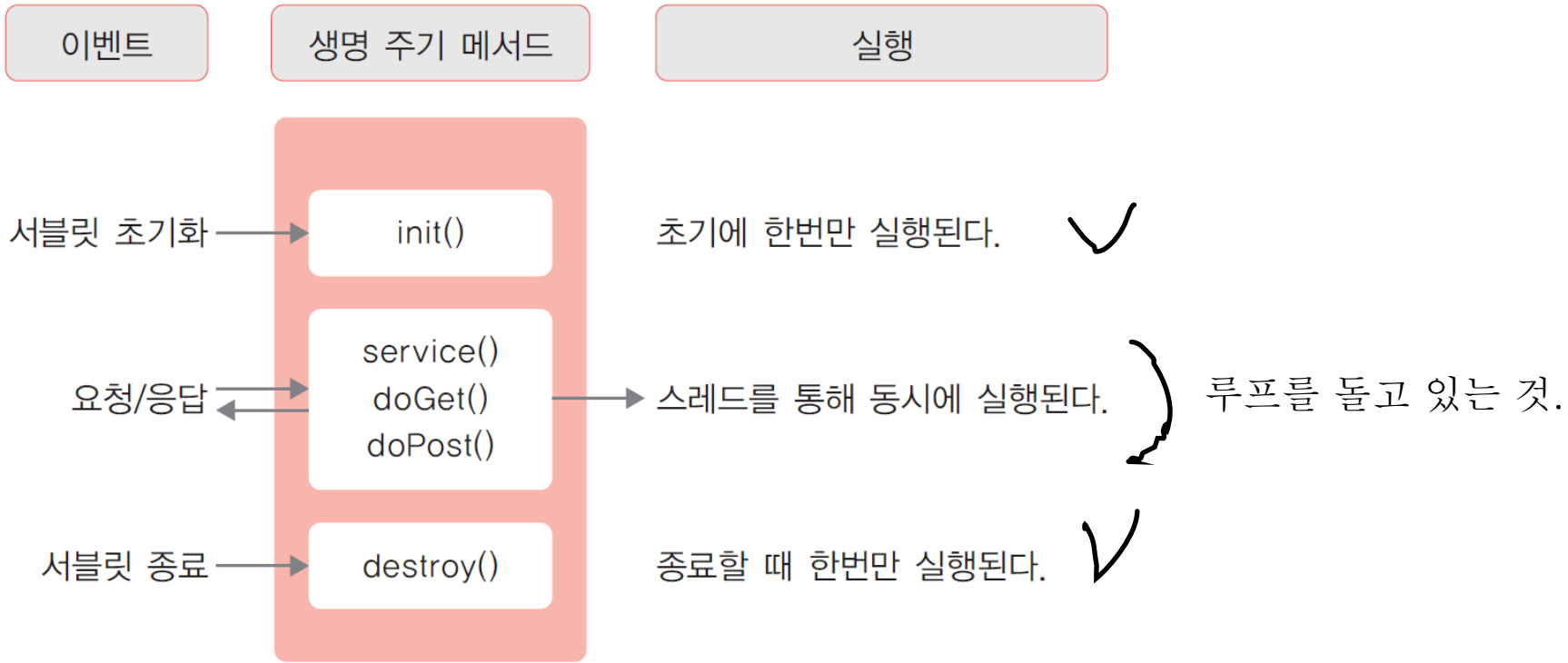


요청이오면 서비스가 호출됨.

서블릿 LifeCycle 메서드

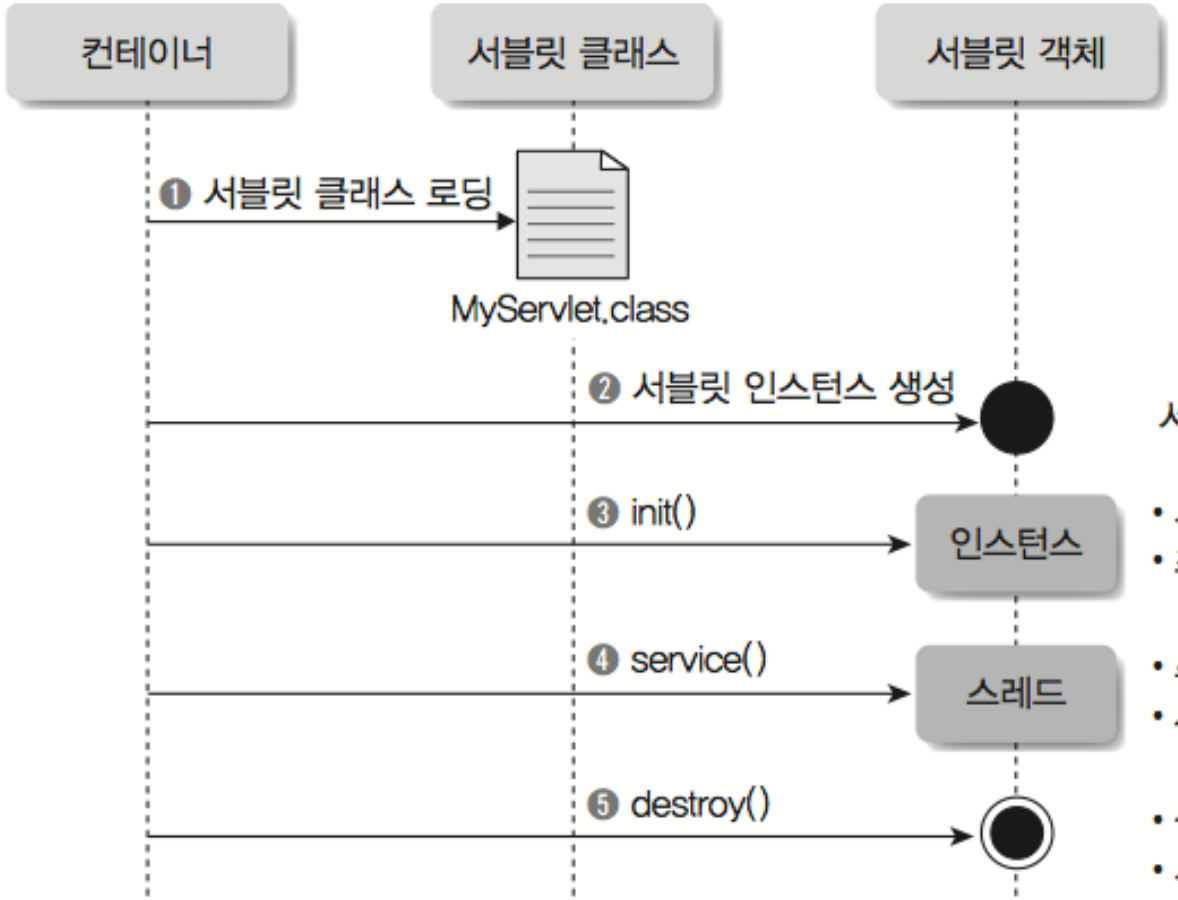
서블릿의 생명 주기

- 객체의 생성에서 종료에 이르는 과정



서블릿 LifeCycle 메서드

서블릿의 동작 과정



서블릿 생성자가 호출된다.

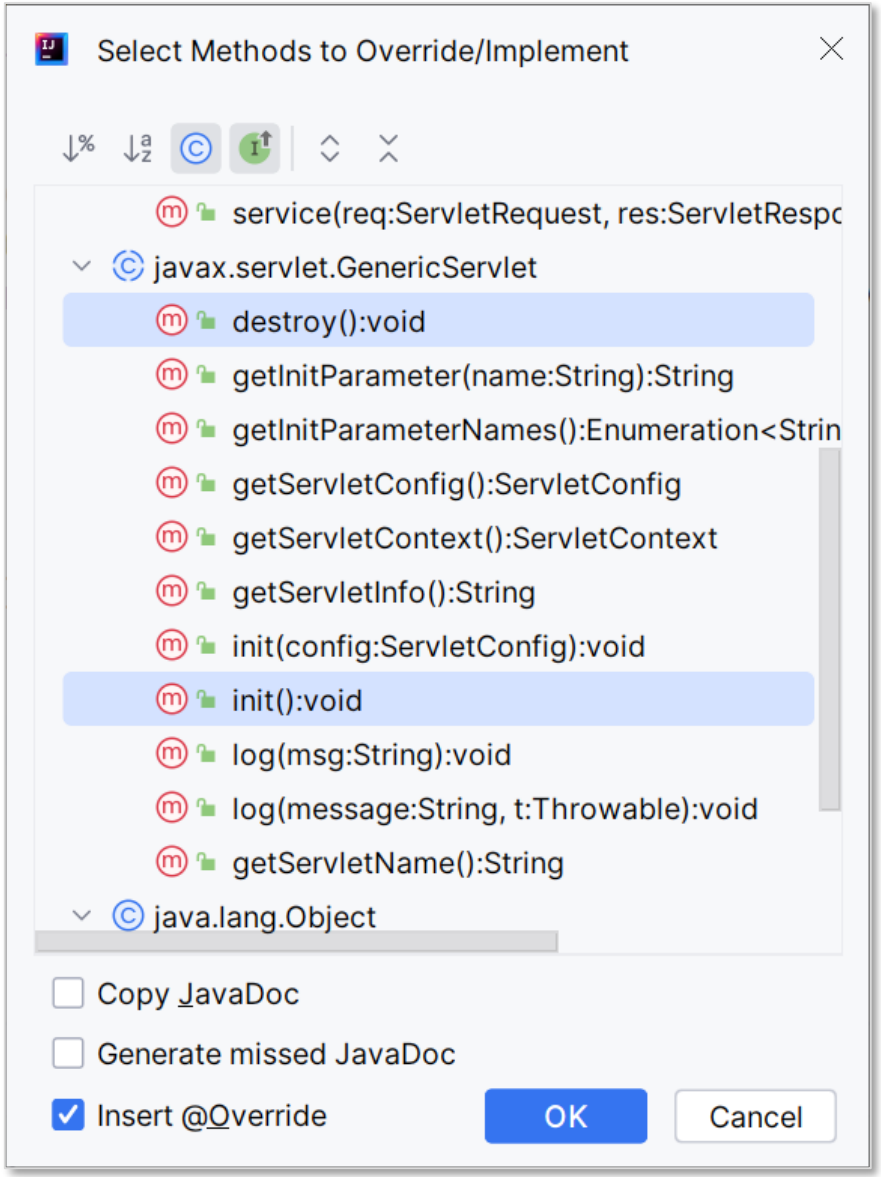
- 서블릿 클래스에서 단 한 번만 실행되는 메서드다.
- 초기화 작업을 수행한다.

- 스레드 사용자 요청을 처리한다.
- 사용자 요청에 따라 service() 메서드가 반복해서 호출된다.

- 상태 종료될 때 호출되는 메서드다.
- 서블릿이 소멸된다.

서블릿 LifeCycle 메서드

- **init, destroy 메서드 추가**
  - HelloServlet.java에서 Ctrl+O
  - init(): void, destroy(): void 선택



부모 클래스를 보면 얻너것을 오버라이드할 수있는지 자세히 볼수있다

서블릿 LifeCycle 메서드

MyServlet

init, destroy 메서드 추가

```
@WebServlet(name="MyServlet", urlPatterns={"/xxx", "/yyy" })
public class MyServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {
        …
    }

    @Override
    public void destroy() {
        System.out.println("destroy 호출");
    }

    @Override
    public void init() throws ServletException {
        System.out.println("init호출");
    }
}
```

서블릿 응답 처리

- 클라이언트에서 서블릿으로 요청
- 서블릿은 처리한 결과를 html 형식으로 응답 처리

톰캣은 운영체제 문자셋을 따름.

- 문자셋 설정

초기에 신경써야함.

맥과 리눅스 사용자는 운영체제 디폴트 문자셋이 UTF-8이기에 신경안서도됨. 윈도우는 utf-8아님.

  - response.setContentType("text/html;charset=UTF-8");
    - 응답 데이터의 MIME 타입

문자셋이 안맞으면
서버와 브라우저에서 데이터가 깨진다.

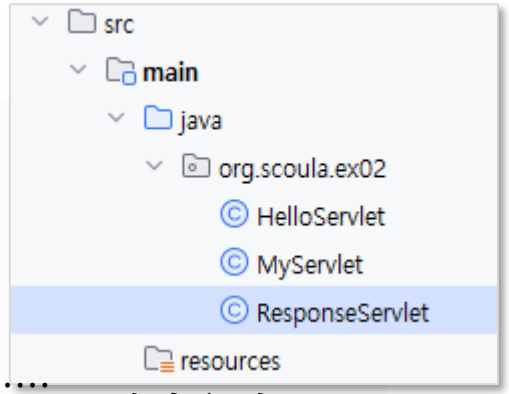
- 응답 데이터의 전송

보통 개발은 윈도우즈에서하고 배포는 리눅스로 하기에
배포할 환경에 맞춰서 개발해야하긴함

  - 문자 데이터 응답
    - response.getWriter()로 PrintWriter 클래스 사용
  - 바이너리 데이터 응답
    - response.getOutputStream()으로 ServletOutputStream 클래스 사용

서블릿 응답 처리

ResponseServlet.java




```
@WebServlet("/response")
public class ResponseServlet extends HttpServlet {

  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                throws IOException, ServletException {

    // MIME 타입 설정
    resp.setContentType("text/html;charset=UTF-8");

    // 자바 I/O
    PrintWriter out = resp.getWriter();

    // html 작성 및 출력
    out.print("<html><body>");
    out.print("ResponseServlet 요청 성공");
    out.print("</body></html>");

  }
}
```

하지만 ㅇㅇ이런 방식이라면 너무 복잡해지는데.....
변수에다가 이스케이프시퀀스사용해야하고 등등등 개같은거
사람이 할짓 못함.
실제 html문서 내용을 어떻게 구성하면 좋을까
=>JSP기술.이 이를 해결해줌!!!
이때 jsp등장.
기본적으로 html형식이지만 vue처럼 변수를 표현하는
java코드가 들어갈 수 있는 범위를 표현하는..

jsp장점 html출력이 쉽다. servelete은 업렵다
serveletdms 자바코드 작성이 쉽다.
jsp는 자바코드표현이 어렵다(추가적인 표현이 필요해서)
=>html작업은 JSP가. 비즈니스로직(javacode)는servlet dl

http://localhost:8080/response

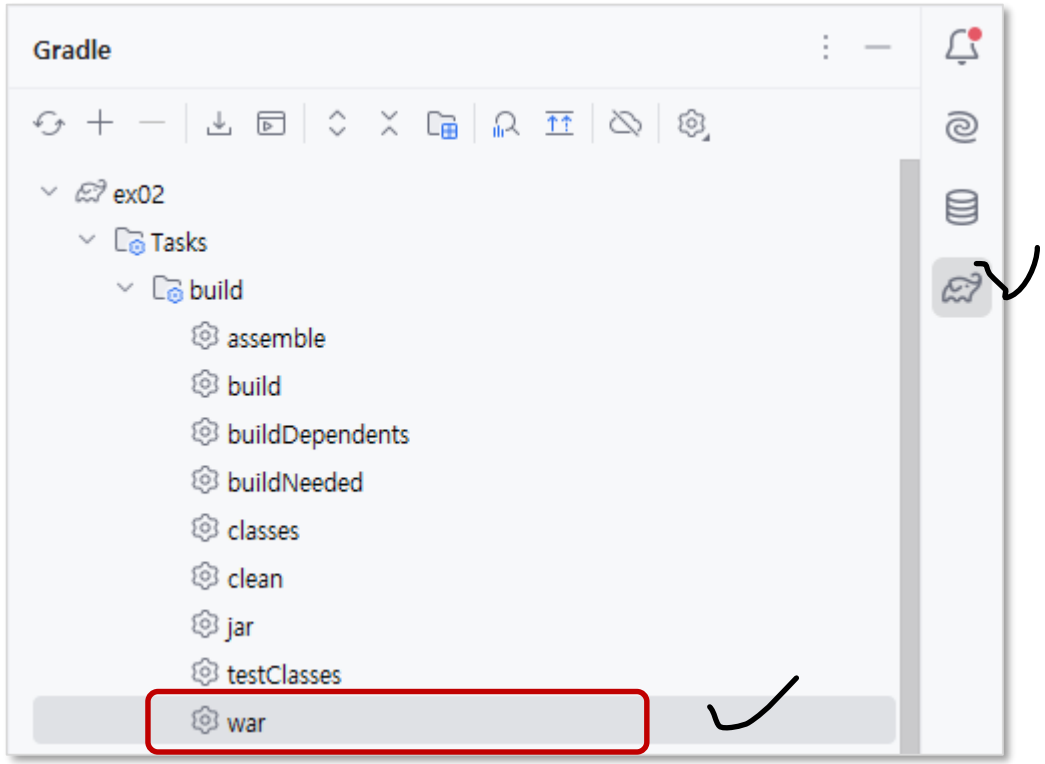
ResponseServlet 요청 성공

실제는 요청을 서블릿이 받고
서블릿이 비즈니스로직으로 결과를 넘기고 종료후 jsp가 html작업을 하는 과정을 거친다.

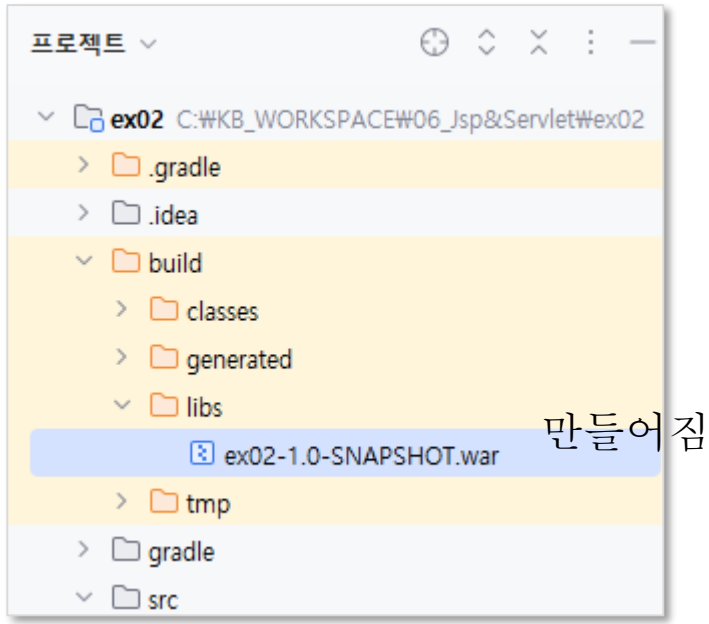
ResponseServlet ?? ??

jsp가 작업후 요청에 응답하는 결과를 보낸다

war 파일 만들기



다블 클릭

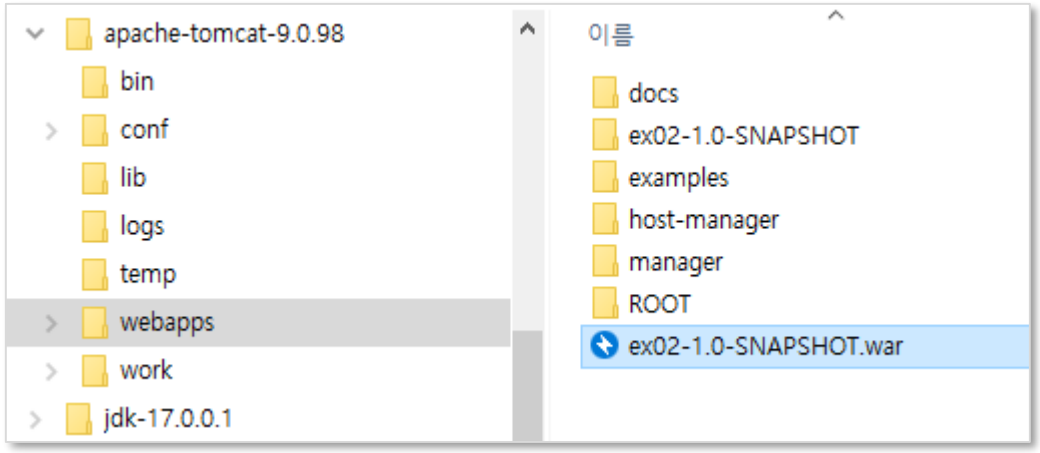


만들어짐

이파일을 옮겨야함

배포하기

war 파일 배치



c://아파치톰캣홈디렉/wepapps/ 위치에 옮겨

ROOT 애플리케이션 설정

C:/apache-tomcat-9.0.98/conf/server.xml

```
    …
      <Host name="localhost"  appBase="webapps" unpackWARs="true" autoDeploy="true">
        …
        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
               prefix="localhost_access_log" suffix=".txt"
               pattern="%h %l %u %t &quot;%r&quot; %s %b" />

        <Context path="/" docBase="ex02-1.0-SNAPSHOT"  reloadable="false" > </Context>
      </Host>
    </Engine>
  </Service>
</Server>
```

war파일의
파일명

배포하기

실제 Tocamcat 서버 실행

- C:₩apache-tomcat-9.0.98₩bin₩startup.bat 실행
  - 윈도우인 경우 기동 될 때 콘솔에 출력되는 한글 깨짐
    실행후 접속해보자
  - http://localhost:8080/ 접속

주의 실행시킬때 intellij에서 구동은 꺼놔야해.
같은 포트임으로

실제 개발 완료된 실제 배포판이 잘 작동하는지 확인

Tomcat 서버 설치

콘솔 한글 로그 문자셋 설정(윈도우 만)

- C:₩apache-tomcat-9.0.89₩conf₩logging.properties

```
...
############################################################
# Handler specific properties.
# Describes specific configuration info for Handlers.
############################################################

…
java.util.logging.ConsoleHandler.encoding = EUC-KR

...
```

실제 배포과정에서
한글 깨질시

→ IntelliJ에서 실행하는 경우 한글 깨짐!!

jsp와 서블릿을 어떻게 붙이는가는 다음 시간에