# Servlet 이해와 활용

egyou@induk.ac.kr
https://cafe.naver.com/induksoft

# 학습 개요

- **학습 배경**
  - JSP와 Servlet는 웹 애플리케이션을 개발하기 위해 도입된 자바 기반 개발 기술이다. Servlet은 웹 디자이너에게는 생소하지만 자바 개발자에게 친숙하며, 자바가 지원하는 모든 API를 사용할 수 있기 때문에 강력한 기능 활용하여 웹 애플리케이션을 개발할 수 있다.
- **학습 목표**
  - Servlet의 등장 배경과 특징을 알아보고, JSP와의 관계를 이해함으로써 웹 애플리케이션 개발에서 Servlet 활용 능력을 배양한다.
- **주요 용어**
  - 서블릿 생명주기, HttpServlet, Deployment Descriptor, Annotation(애노테이션)

# 학습 내용

- **Servlet의 정의, 등장 배경, 특징에 대하여 알아본다.**
  - 웹 서비스 처리 과정에 대하여 알아본다
- **Servlet의 구조와 생명주기에 대하여 알아본다.**
- **Servlet을 이용한 웹 요청 처리 과정에 대하여 알아본다.**
- **Servlet과 JSP의 특징을 살펴보고, 비교해본다.**

# 웹 서비스 처리 과정

1. 사용자가 웹 브라우저 주소창에 URL입력
2. 웹 브라우저가 HTTP GET 또는 HTTP POST 요청으로 생성하고, 웹 서버에게 요청 전송
3. 웹 서버는 요청한 페이지를 찾고, HTTP 응답을 작성하고, 웹 브라우저에 전송
4. 웹 브라우저는 HTTP 응답을 수신하여 콘텐츠를 화면에 표시함
   - HTTP 응답은 응답 헤더(상태 코드, 컨텐츠 타입 ...) 컨텐츠로 구성됨

# 계속

- **3단계에서 요청이 새로운 페이지 생성이나 서버에 자료 저장인 경우**
  - 웹 서버가  웹 컨테이너에게 요청을 전송하면, 웹 컨테이너가 페이지를 생성하고, 웹 서버에게 전송
  - 웹 서버는 HTTP 응답으로 작성하고, 웹 브라우저에 전송
- **웹 서버가 요청한 페이지를 찾지 못하는 경우**
  - 404 File Not Found 응답을 웹 브라우저에 전송

# Servlet 정의와 등장 배경

## ▪정의

- 요청-응답 프로그래밍 모델 방식으로 접근되는 응용 프로그램들을 제공하는 서버의 유용성을 증대하기 위해 사용되는 Java 언어로 작성된 클래스를 의미하며, 일반적으로 Java Servlet API를 준수하는 자바 엔터프라이즈 에디션(Java EE)에 포함된 자바 클래스들을 의미한다.

# 계속

- **등장 배경**
  - CGI 한계를 극복하기 위해 개발되었다.
  - 프로세스 기반으로 동작하기 때문에 성능이 저하된다.
  - 개발 언어에 따라 이식성 문제가 발생한다.
  - 자바 플랫폼에서 컴포넌트를 기반으로 한 웹 애플리케이션 개발에 대한 요구 증가하였다.

# 계속

- CGI 방식으로 개발한 웹 애플리케이션은 웹 서버가 요청하면 개별 프로세스(process)를 생성하여 이를 처리
  - 프로세스는 간단하게 실행 중인 프로그램
  - 하나의 프로세스는 코드, 정적 데이터, 스택, 힙 등과 같은 메모리 구성
  - 실행 코드뿐 아니라 데이터들의 중복으로 인하여 메모리 사용이 크게 증가하여 시스템 성능 저하를 유발

# 계속

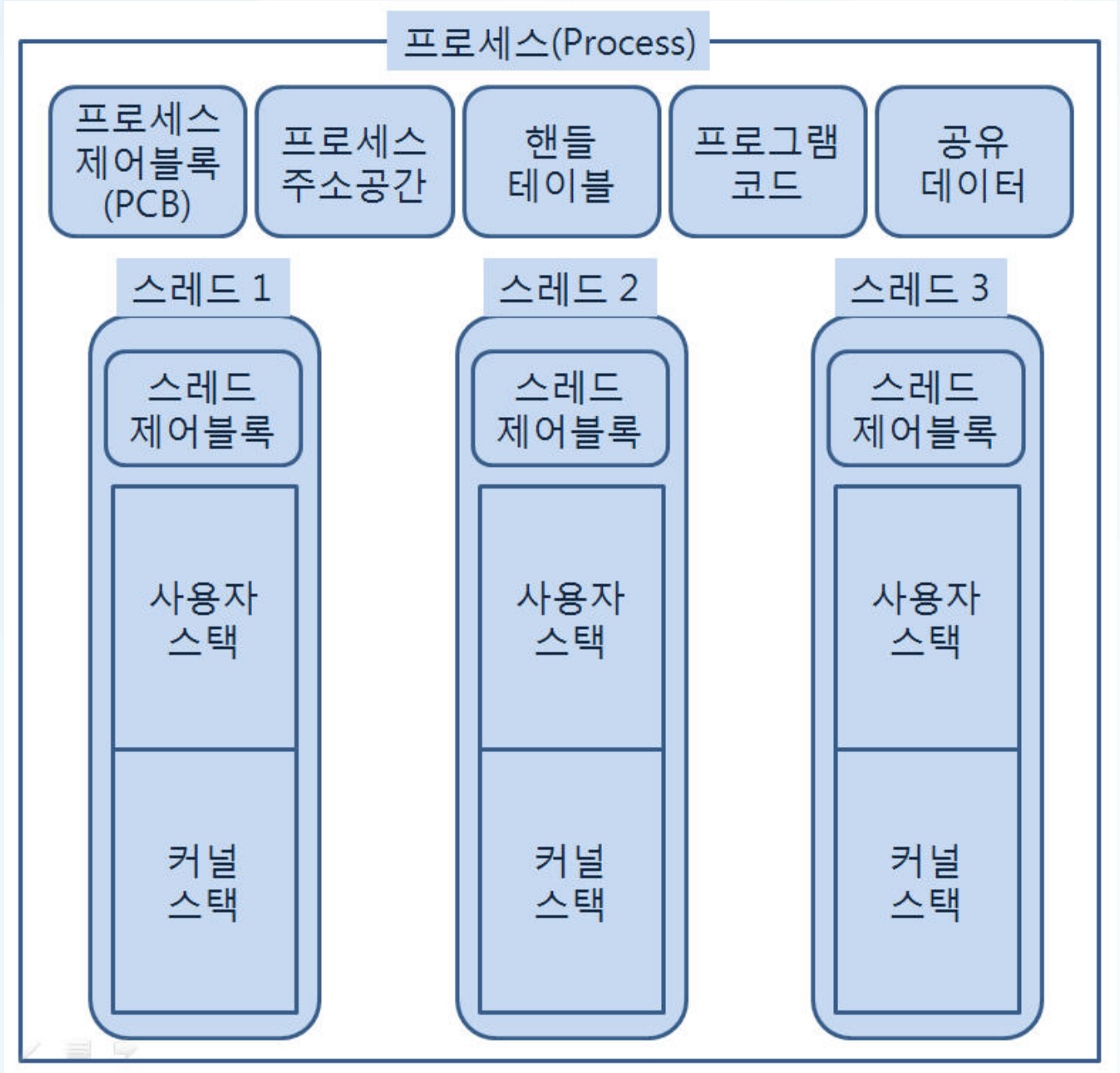
- 개발 언어에 따라 이식성 문제가 발생
  - C로 작성된 윈도우용 CGI 프로그램은 리눅스나 맥에서는 실행되지 않기 때문에 다시 컴파일을 하고, 배포해야 한다는 문제
  - 웹 페이지의 구성요소들 간의 구조를 정의하는데 사용되는 HTML 처리가 쉽지 않고, 유지보수가 어려움

# 계속

- **특징**
  - 확장이 용이하고 플랫폼 독립적인 웹 응용프로그램 개발이 가능함
  - 스레드 기반 -> 높은 효율성 제공
  - 자바 기반 -> 자바 API를 모두 사용할 수 있음
  - 운영체제, 하드웨어 독립적인 개발 및 운영 환경 구축 가능

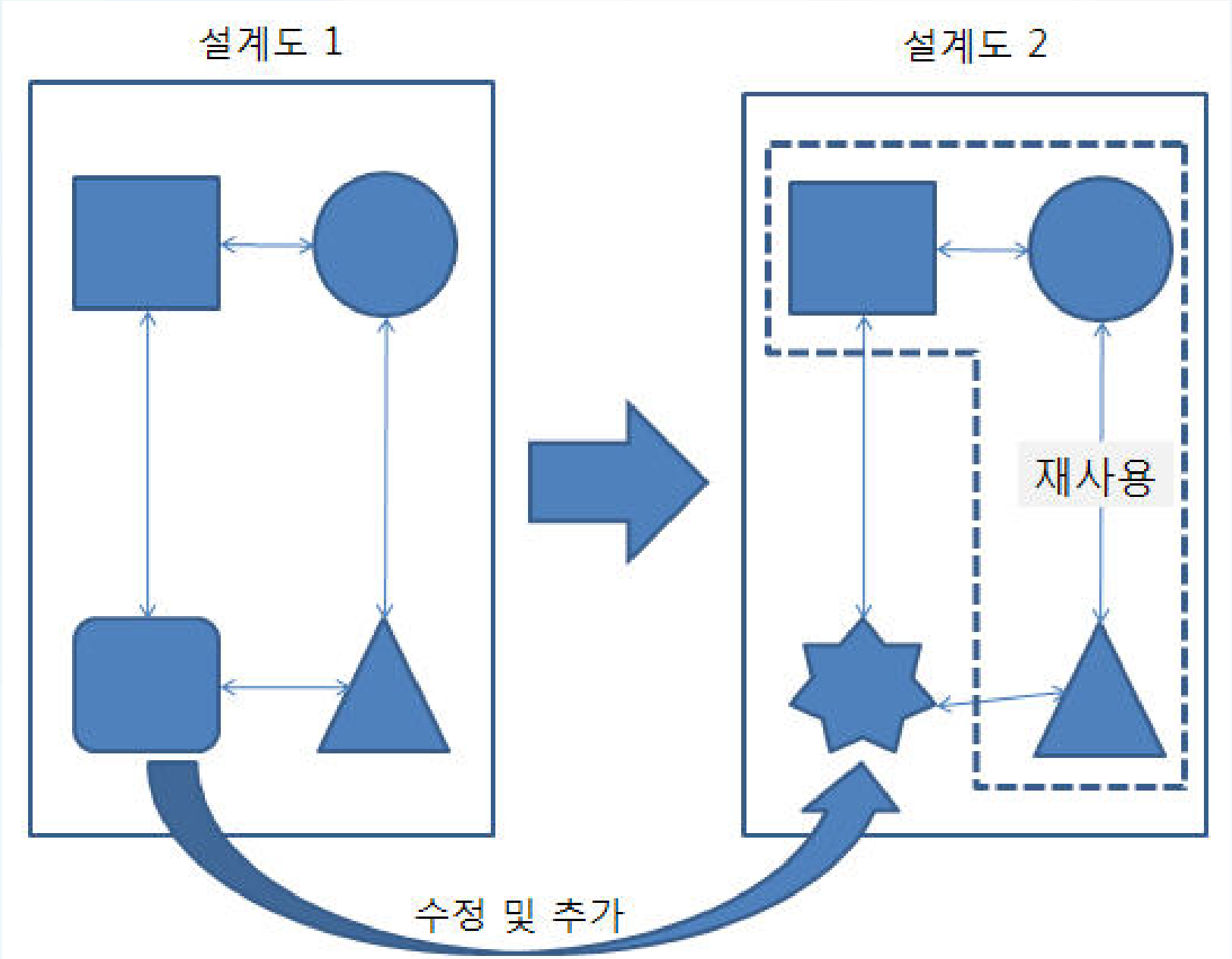
# 계속

- 멀티스레드 기반
  - 스레드란 가장 작은 처리 단위, 일반적으로 프로세스의 부분집합(subset)으로 존재한다.
  - 멀티스레드(multithread)는 하나의 프로세스안에 다수의 스레드가 존재하는 것을 의미한다. 같은 프로세스에 포함된 스레드들은 메모리, 상태 정보 등 다양한 자원을 공유하며, 주소 공간을 공유한다. 또한 스레드 간 통신이 용이하고, 스레드간 신속한 컨텍스트 스위칭(context switching)이 가능


프로세스(Process)
프로세스 제어블록 (PCB)
프로세스 주소공간
핸들 테이블
프로그램 코드
공유 데이터
스레드 1
스레드 제어블록
사용자 스택
커널 스택
스레드 2
스레드 제어블록
사용자 스택
커널 스택
스레드 3
스레드 제어블록
사용자 스택
커널 스택

# 계속

- 자바 기반
  - 플랫폼에 독립적이며, 이식성 문제를 발생하지 않으며 높은 생산성과 신뢰성을 기대할 수 있다.
  - 또한 자바 API를 모두 사용할 수 있기 때문에 강력하고 다양한 기능을 제공
  - 객체지향 프로그래밍 지원
    - 설계도와 제작 공정을 통해 동일한 품질의 부품을 다량으로 생산
    - 특징과 기능을 가진 프로그래밍 요소들로 구성되어 있으며 상호작용을 통해 주어진 문제를 해결


설계도 1
설계도 2
재사용
수정 및 추가

# 객체지향 프로그래밍

## ▪주요 요소

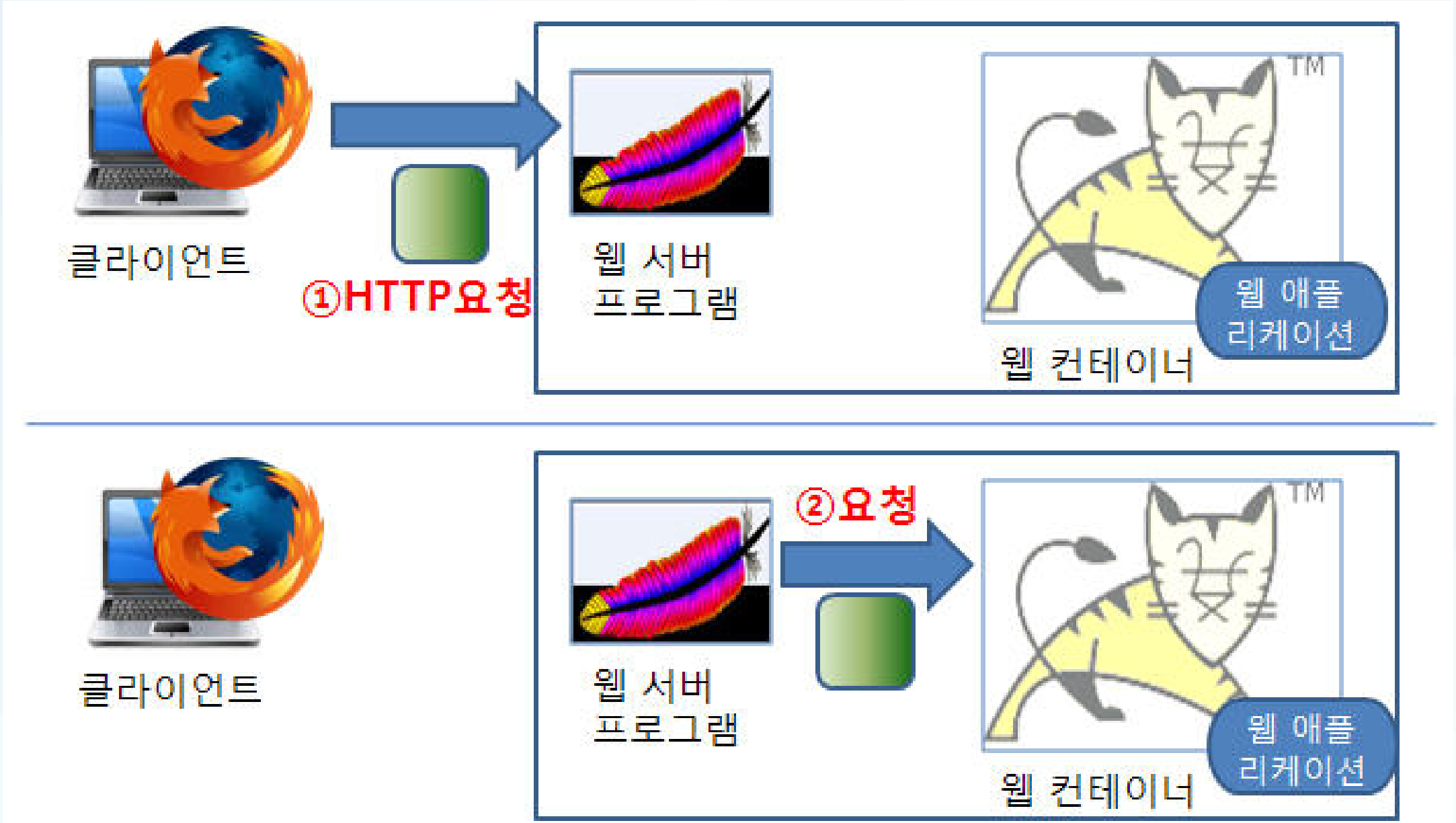
- 객체(object)는 특징과 기능을 갖고 실제로 동작하는 프로그래밍 요소
- 클래스(class)는 객체를 효율적으로 생성하기 위해서 객체에 대한 정의를 수행하는 프로그래밍 요소
- 추상 클래스(abstract method)는 멤버 변수, 멤버 메서드, 그리고 하나 이상의 구현이 완성되지 않은 메서드, 즉 추상 메서드를 갖는 클래스
- 인터페이스(interface)는 멤버 변수들로만 구성되거나, 멤버 변수와 추상 메서드들로만 구성되며, 다중 상속(multiple inheritance)이 가능

# 객체지향 프로그래밍

- **주요 특징**
  - 캡슐화(encapsulation)
    - 객체의 속성과 행위를 묶어서 하나의 요소로 처리하여 유지보수를 용이하게 하고, 구현에 관한 상세 사항을 외부에 감추는 기능을 함께 제공한다.
  - 상속성(inheritance)
    - 두 클래스들 간의 관계로 기존의 클래스를 기반으로 하여 새로운 클래스를 정의하는 기능으로 높은 생산성과 신뢰성 있는 클래스를 정의할 수 있도록 한다.
  - 다형성(polymorphism)
    - 객체의 종류에 따라 다른 연산을 수행하도록 하는 기능으로 프로그램의 확장성 및 유지보수성 증대를 제공한다.

# 서블릿 동작 과정

- ① 단계 : 사용자가 웹 브라우저를 이용하여 요청을 하면 웹 서버는 요청에 해당하는 정적인 웹 페이지를 찾는다.
- ② 단계 : 정적인 웹 페이지가 존재하지 않는 경우 웹 컨테이너에게 동적인 웹 페이지를 생성하도록 요청 정보를 전송한다.


클라이언트
①HTTP요청
웹 서버
프로그램
웹 애플
리케이션
웹 컨테이너
클라이언트
웹 서버
프로그램
②요청
웹 애플
리케이션
웹 컨테이너

# 서블릿 동작 과정

- ③ 단계 : 웹 컨테이너는 “HttpServletRequest” 객체와 “HttpServletResponse” 객체를 생성한다.
- ④ 단계 : 웹 컨테이너는 해당 서블릿 객체로부터 서블릿 스레드를 생성하며, 이 때 생성된 요청, 응답 객체를 매개변수로 전달한다. 만약 서블릿 객체가 존재하지 않는 경우 서블릿 클래스를 메모리에 적재하고, 서블릿 객체를 생성한다.

클라이언트

웹 서버
프로그램


웹 컨테이너

클라이언트

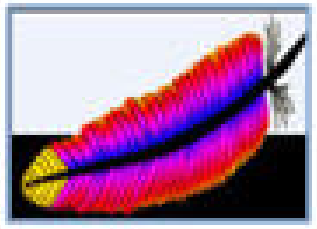
웹 서버
프로그램


웹 컨테이너

# 계속

- ⑤ 단계 : 웹 컨테이너는 서블릿 스레드의 service() 메서드를 호출하고, 요청한 HTTP 메서드를 처리할 수 있는 메서드를 결정한다.
- ⑥ 단계 : 요청한 HTTP 메서드를 실행하여 동적인 웹 페이지를 생성한 후, 이를 응답 객체에 저장한다.

클라이언트



웹 서버
프로그램

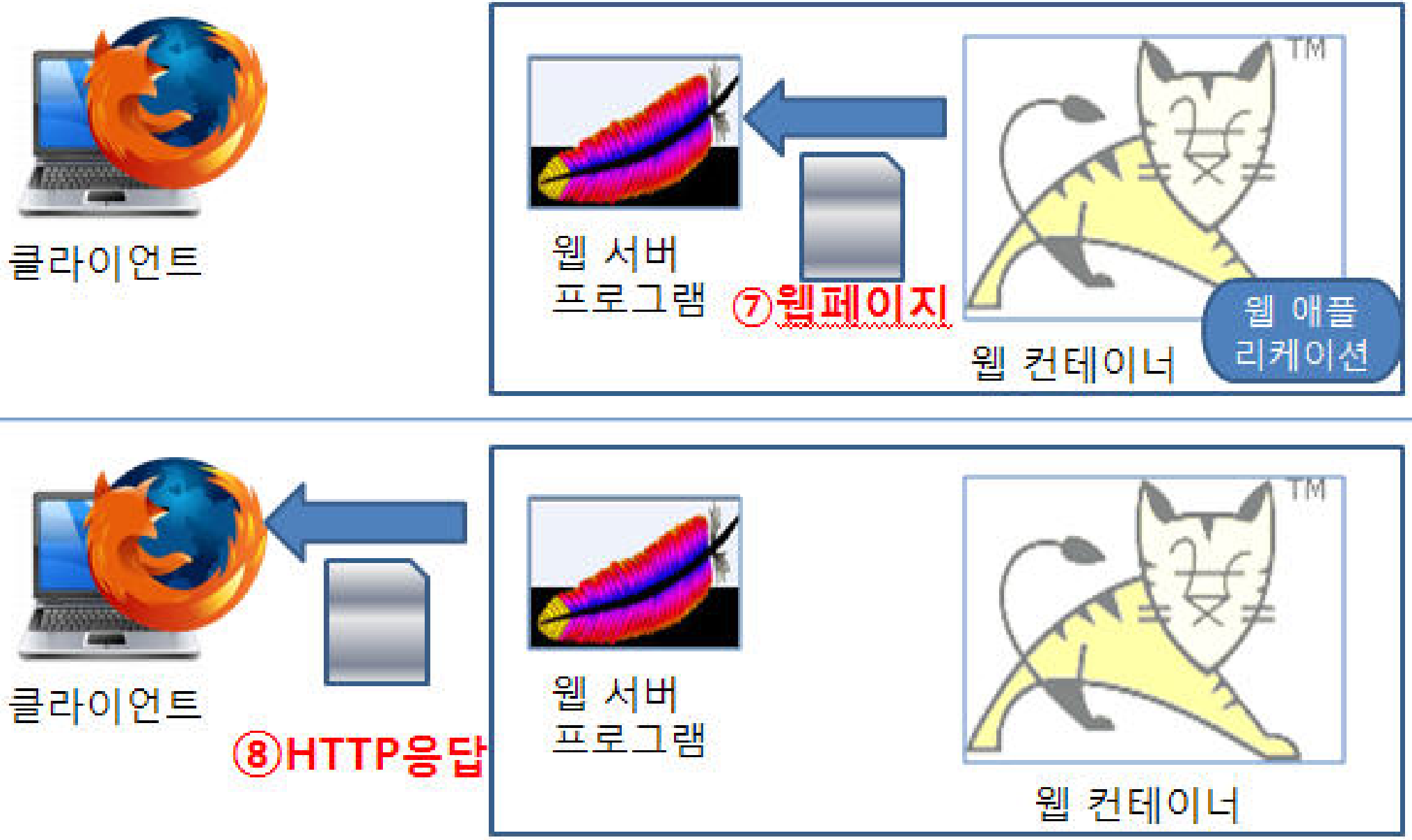

클라이언트

웹 서버
프로그램

# 계속

- ⑦ 단계 : 웹 컨테이너는 동적인 웹 페이지, 컨텐츠를 웹 서버에서 전송한다.
- ⑧ 단계 : 웹 서버는 동적으로 생성된 웹 페이지, 컨텐츠에 응답 헤더를 추가하여 HTTP 응답을 작성하고 이를 웹 브라우저에게 전송한다.

- 웹 브라우저는 전송받은 HTTP 응답을 분석하고, 화면에 표시한다.

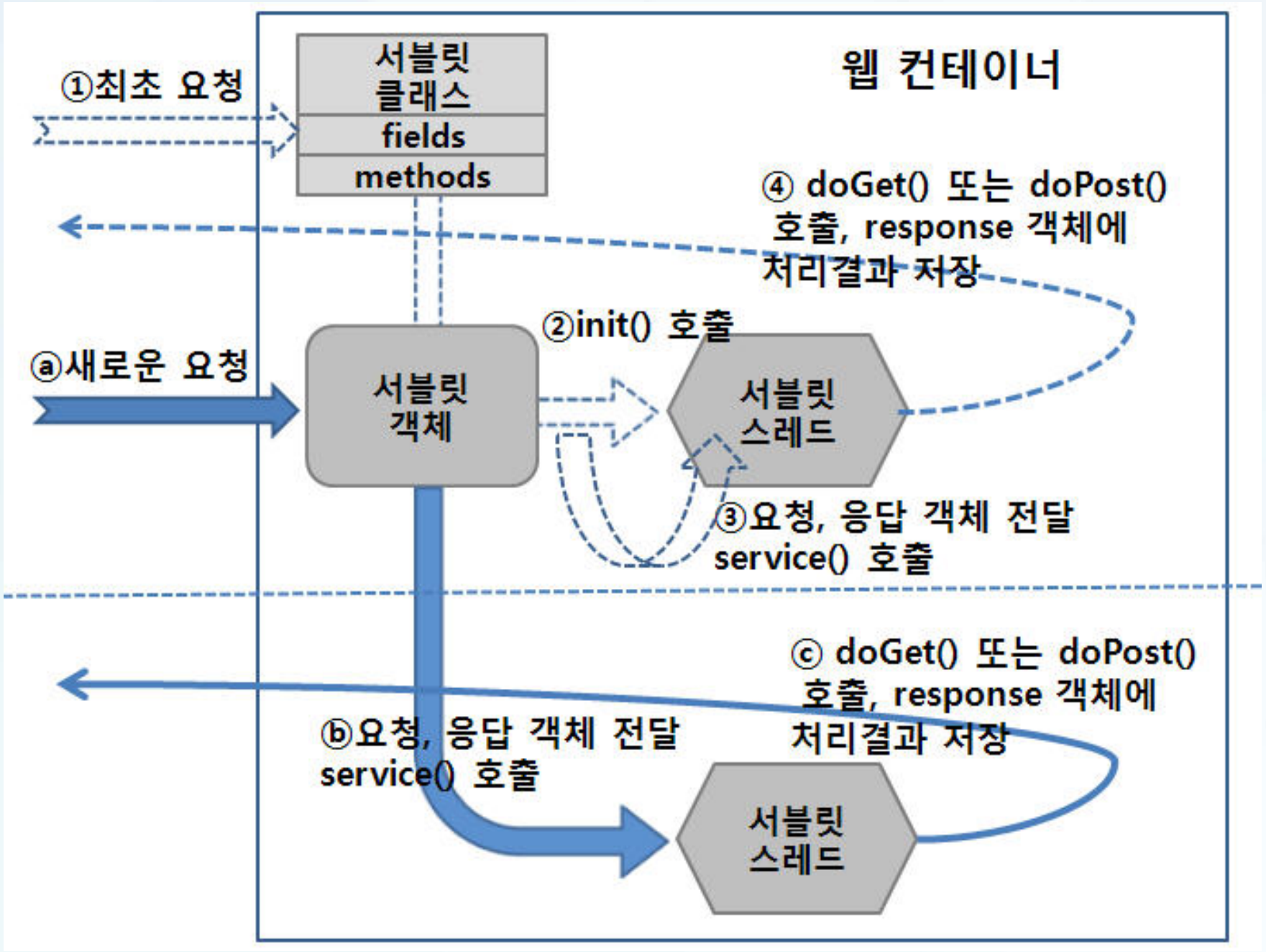

클라이언트
웹 서버
프로그램
⑦웹페이지
웹 컨테이너
웹 애플
리케이션
⑧HTTP응답
클라이언트
웹 서버
프로그램
웹 컨테이너

# Servlet을 알아야 하는 추가적인 이유

- **JavaEE의 기반이 되는 기술**
- **JSP 표준의 기반이 되는 기술**
  - JSP는 주로 사용자에게 결과를 보여주는 프레젠테이션 로직을 개발하기 위해 사용된다.
    - 단순한 프레젠테이션의 경우 템플릿 엔진(Thymeleaf, Mustache, Groovy, Freemarker) 등을 사용할 수 있음
  - 서블릿은 사용자의 요청을 받아서 처리하는 비즈니스 로직을 개발하기 위해 사용된다.

# 서블릿 생명 주기

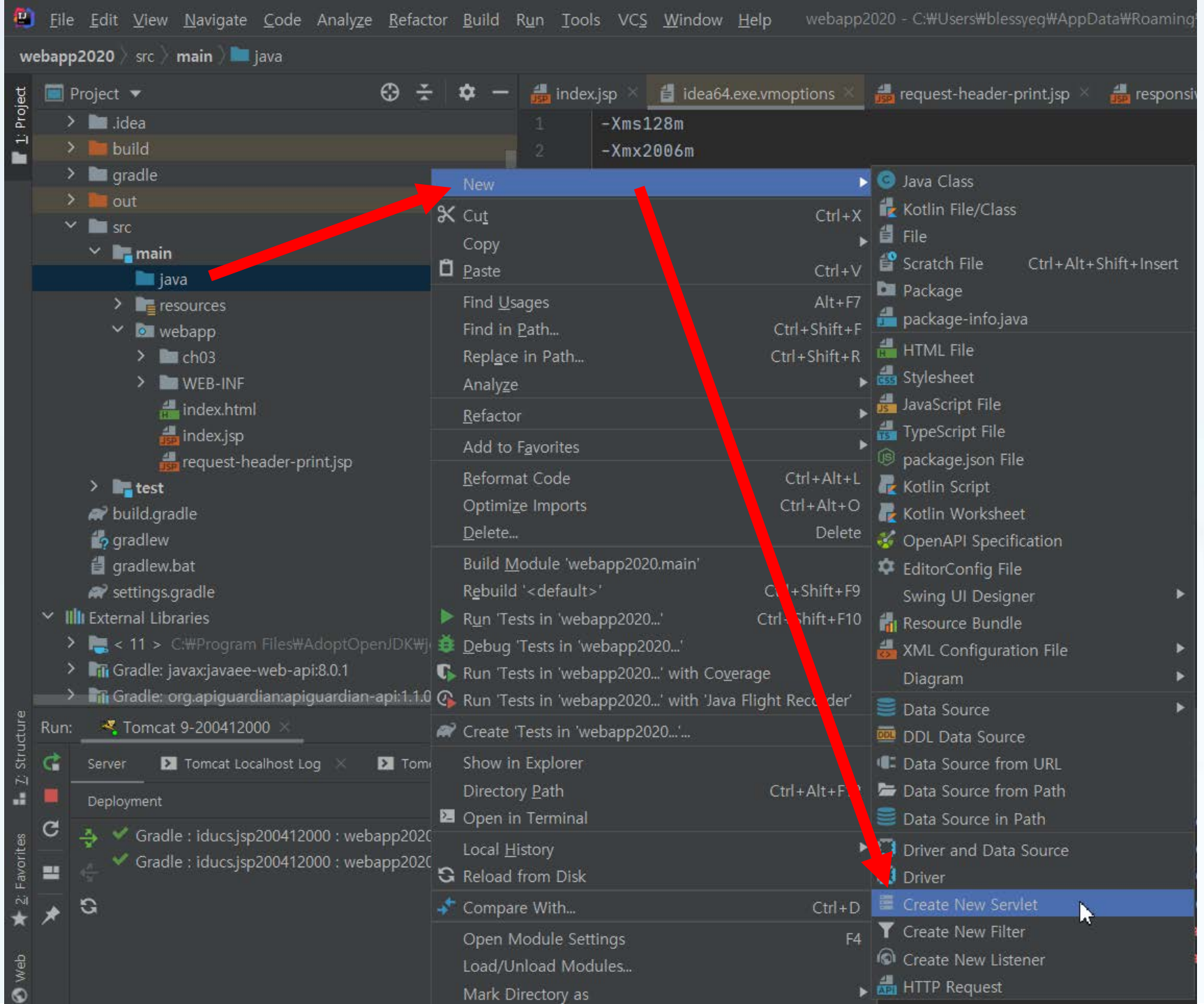
- ① 서블릿 클래스 파일을 메모리에 적재(loading)하고, 생성자를 호출하여 인스턴트 생성
- ② 인스턴스화된 서블릿 객체의 초기화를 위해 init() 메서드 호출
- ③ 요청마다 새로운 서블릿 스레드 생성하고, service() 메서드 호출
- ④ 요청한 HTTP 메서드에 따라 doGet() 또는 doPost() 등을 호출
- ⑤ 서블릿 소멸화를 위해 destroy() 메서드 호출
  - 응답이 필요없거나 컨테이너로부터 종료 요청을 받은 경우 자원 정리 등 추가 작업을 처리한다.

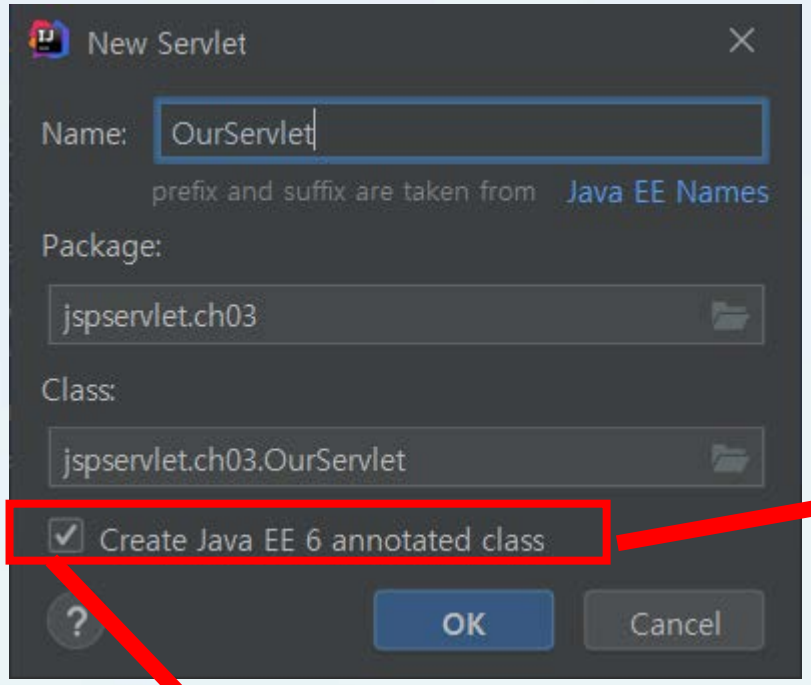

웹 컨테이너
서블릿
클래스
fields
methods
①최초 요청
④ doGet() 또는 doPost()
호출, response 객체에
처리결과 저장
②init() 호출
ⓐ새로운 요청
서블릿
객체
서블릿
스레드
③요청, 응답 객체 전달
service() 호출
ⓒ doGet() 또는 doPost()
호출, response 객체에
처리결과 저장
ⓑ요청, 응답 객체 전달
service() 호출
서블릿
스레드

# 프로젝트에 Servlet 추가

- **Intellij**
  - src > main > java 에서 New > Create New Servlet

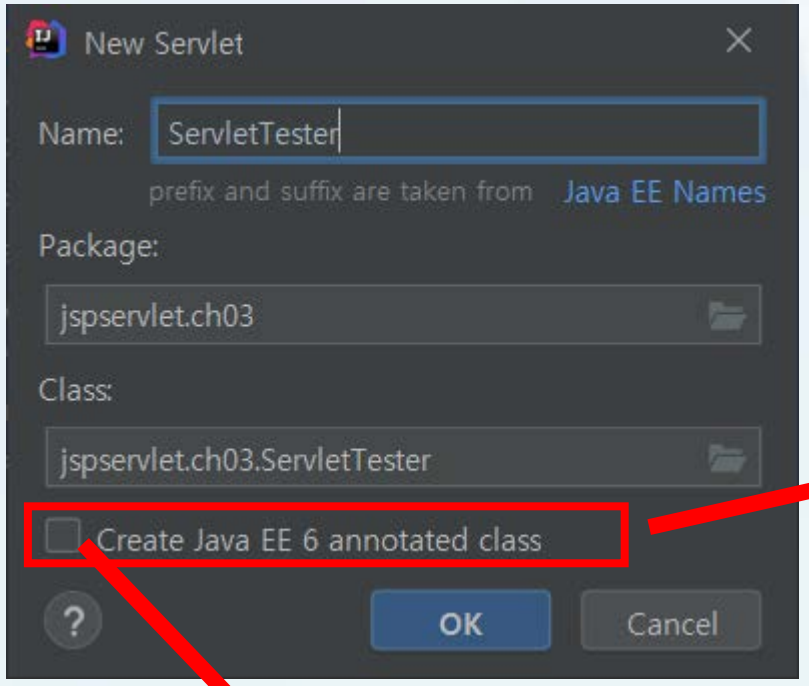
- **Eclipse**
  - Java Resources > src > 에서 New > Servlet


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
webapp2020 - C:₩Users₩blessyeg₩AppData₩Roaming
webapp2020 src main java
Project
.idea
build
gradle
out
src
main
java
resources
webapp
ch03
WEB-INF
index.html
index.jsp
request-header-print.jsp
test
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Gradle: javax:javaee-web-api:8.0.1
index.jsp
idea64.exe.vmoptions
request-header-print.jsp
-Xms128m
-Xmx2006m
New
Cut Ctrl+X
Copy
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Build Module 'webapp2020.main'
Rebuild '<default>' Ctrl+Shift+F9
Run 'Tests in 'webapp2020...' Ctrl+Shift+F10
Debug 'Tests in 'webapp2020...'
Run 'Tests in 'webapp2020...' with Coverage
Create 'Tests in 'webapp2020...'...
Show in Explorer
Directory Path Ctrl+Alt+F
Open in Terminal
Local History
Reload from Disk
Compare With... Ctrl+D
Open Module Settings F4
Load/Unload Modules...
Mark Directory as
Java Class
Kotlin File/Class
File
Scratch File Ctrl+Alt+Shift+Insert
Package
package-info.java
HTML File
Stylesheet
JavaScript File
TypeScript File
package.json File
Kotlin Script
Kotlin Worksheet
OpenAPI Specification
EditorConfig File
Swing UI Designer
Resource Bundle
XML Configuration File
Diagram
Data Source
DDL Data Source
Data Source from URL
Data Source from Path
Data Source in Path
Driver and Data Source
Driver
Create New Servlet
Create New Filter
Create New Listener
HTTP Request
Run: Tomcat 9-200412000
Server
Tomcat Localhost Log
Deployment
Gradle : iducs.jsp200412000 : webapp202
Gradle : iducs.jsp200412000 : webapp202




```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns
         version="4.0">
</web-app>
```

```java
package jspservlet.ch03;

import ...

@WebServlet(name = "OurServlet")
public class OurServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }

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

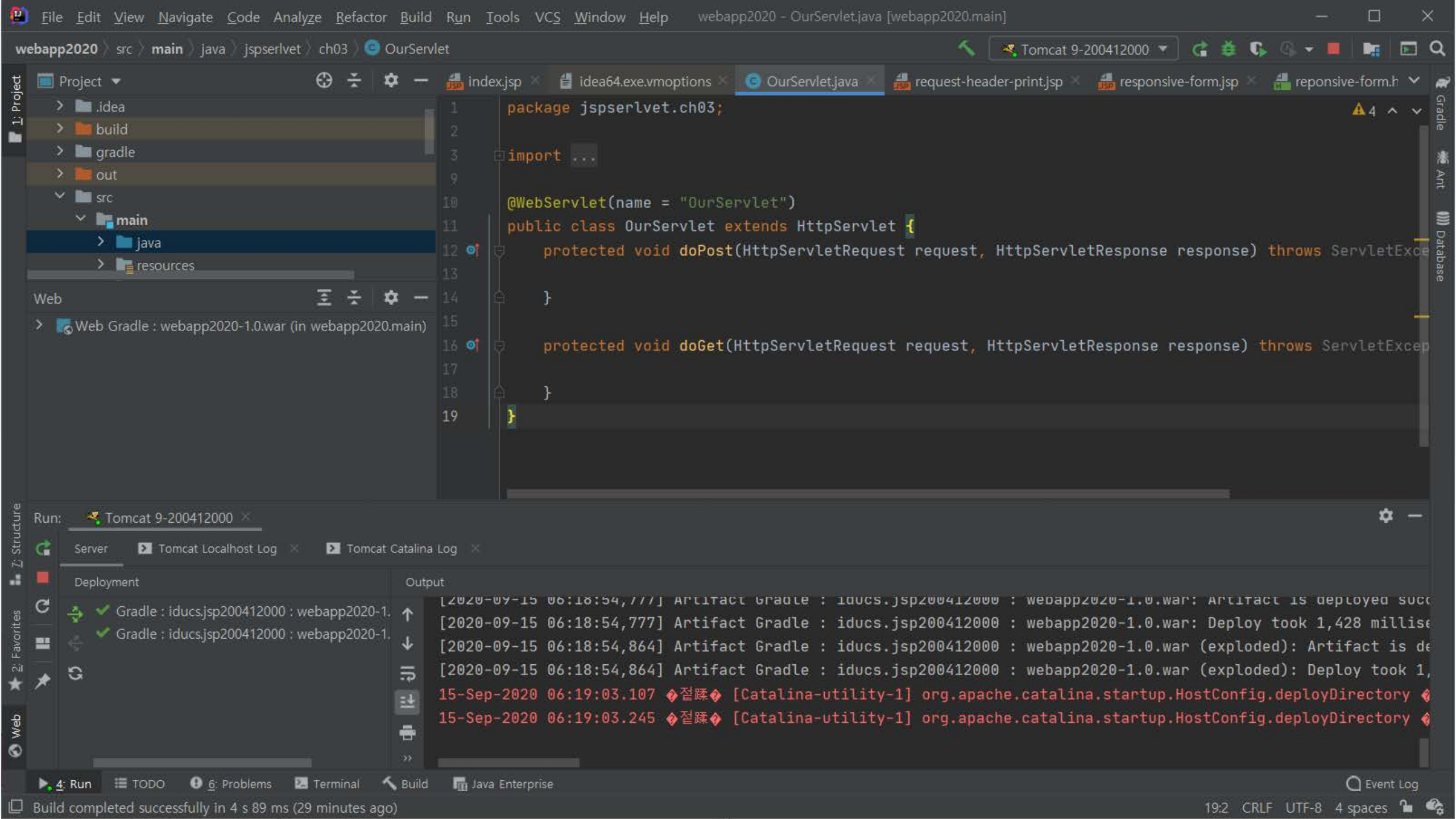



```xml
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         version="4.0">
    <servlet>
        <servlet-name>Servlet</servlet-name>
        <servlet-class>jspserlvet.ch03.Servlet</servlet-class>
    </servlet>
</web-app>
```

```java
package jspserlvet.ch03;

import ...

public class Servlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }
}
```

# 계속

- **서블릿 등록**
  - 요청 URL과 Servlet을 매핑하는 것을 의미함
- **방법**
  - Servlet 3.0 이상은 Annotation(애노테이션)을 이용하여 편리한 처리가 가능함
    - import javax.servlet.annotation.WebServlet;
    - @WebServlet("/OurServlet")
  - Servlet 3.0 이전에는 WEB-INF₩web.xml을 이용하여 등록해야 함


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
webapp2020 - OurServlet.java [webapp2020.main]
webapp2020 src main java jspserlvet ch03 OurServlet
Tomcat 9-200412000
Project
.idea
build
gradle
out
src
main
java
resources
Web
Web Gradle : webapp2020-1.0.war (in webapp2020.main)
index.jsp
idea64.exe.vmoptions
OurServlet.java
request-header-print.jsp
responsive-form.jsp
reponsive-form.h
package jspserlvet.ch03;
import ...
@WebServlet(name = "OurServlet")
public class OurServlet extends HttpServlet {
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletExce
}
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletExcep
}
}
Run: Tomcat 9-200412000
Server
Tomcat Localhost Log
Tomcat Catalina Log
Deployment
Gradle : iducs.jsp200412000 : webapp2020-1.
Gradle : iducs.jsp200412000 : webapp2020-1.
Output
[2020-09-15 06:18:54,777] Artifact Gradle : iducs.jsp200412000 : webapp2020-1.0.war: Deploy took 1,428 millise
[2020-09-15 06:18:54,864] Artifact Gradle : iducs.jsp200412000 : webapp2020-1.0.war (exploded): Artifact is de
[2020-09-15 06:18:54,864] Artifact Gradle : iducs.jsp200412000 : webapp2020-1.0.war (exploded): Deploy took 1,
15-Sep-2020 06:19:03.107 [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployDirectory
15-Sep-2020 06:19:03.245 [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployDirectory
4: Run
TODO
6: Problems
Terminal
Build
Java Enterprise
Event Log
Build completed successfully in 4 s 89 ms (29 minutes ago)
19:2 CRLF UTF-8 4 spaces

OurServlet.java

```
package jspservlet.ch03;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebInitParam;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(
        urlPatterns = {
                "/OurServlet",
                "/today"
        },
        initParams = {
                @WebInitParam(name = "init-param", value = "init-value")
        })
```

OurServlet.java

```
public class OurServlet extends HttpServlet {
    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init();
        System.out.println("init : " + config.getInitParameter("init-param"));
    }
    public OurServlet() {
        System.out.println("constructor : ");
    }
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        System.out.println("service : ");
        super.service(req, resp);
    }
```

OurServlet.java

```
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) {
        System.out.println("doPost : " + request.getParameter("name") + " : " +
request.getParameter("phone"));
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
{
        System.out.println("doGet : " + request.getParameter("name") + " : " +
request.getParameter("phone"));
    }
}
```

form-get-post.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: blessyeg
  Date: 2020-09-15
  Time: 오전 8:57
  To change this template use File ¦ Settings ¦ File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8"
language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
```

form-get-post.jsp

```
Get Method
<form action="OurServlet" method="get" name="get">
    <label for="name">First name:</label><br>
    <input type="text" id="get-name" name="name" value="egyou"><br>
    <label for="phone">Phone:</label><br>
    <input type="text" id="get-phone" name="phone" value="02-950-7625"><br><br>
    <input type="submit" value="Get">
</form>

Post Method
<form action="OurServlet" method="post" name="post">
    <label for="name">First name:</label><br>
    <input type="text" id="post-name" name="name" value="comso"><br>
    <label for="phone">Phone:</label><br>
    <input type="text" id="post-phone" name="phone" value="02-950-7620"><br><br>
    <input type="submit" value="Post">
</form>
</body>
</html>
```

# Servlet 과 JSP 관계

- **Servlet 장점**
  - Servlet은 매우 우수한 성능을 제공하는 웹 애플리케이션 개발 기술이고, 자바 API를 사용할 수 있기 때문에 강력한 기능을 제공한다.
- **Servlet 단점**
  - HTML 사용이 어렵다.
  - HTML 페이지를 String 객체 참조 변수에 저장하려면 "기호를 ₩"로 변환해야 한다. 그러나 속성의 값은 "기호를 사용하기 때문에 변환해야할 기호의 수가 많을 뿐 아니라, 하나만 실수를 하더라도 컴파일 오류를 발생한다.
- **Servlet 주요 활용**
  - MVC 모델 2에서 컨트롤러 기능을 작성하는데 활용된다.

# 계속

- **Servlet의 HTML 표현상의 문제로 인한 개발과 관리 측면의 문제가 존재한다.**
  - 콘텐츠와 비즈니스 로직이 하나의 소스에 존재하는 경우 수정 및 유지보수가 어려움
  - 웹 디자이너 : Servlet 코드에 대한 이해가 필요함
  - 개발자 : HTML 소스에 대한 이해가 필요함
- **JSP의 특징**
  - HTML에서 비즈니스 로직을 처리할 수 있도록 함
  - 콘텐츠와 비즈니스 로직을 구분할 수 있음

# JSP 동작

- **JSP는 컨테이너에 의해 Servlet으로 변경되고, Servlet으로 동작한다.**
  - 개발자가 index.jsp 작성
  - 컨테이너가 Servlet 파일 index_jsp.java로 변경하고, Servlet 클래스 index_jsp.class로 컴파일
  - 이후 과정은 Servlet의 생명 주기 및 동작과 동일함
- **JSP에서 변환된 Servlet 파일 경로**
  - 톰캣 서버의 /work/Catalina/localhost/server/org/apache/jsp
- **Servlet은 <project이름>/WEB-INF/classes**

# JSP 동작

- **JSP는 컨테이너에 의해 Servlet으로 변경되고, Servlet으로 동작한다.**
  - 개발자가 helloToday.jsp 작성
  - 컨테이너가 Servlet 파일 helloToday_jsp.java로 변경하고, Servlet 클래스 helloToday_jsp.class로 컴파일한다.
  - 이후 과정은 Servlet의 생명 주기 및 동작과 동일함
  - 변환된 Servlet 파일 경로
    - <eclipse workspace 디렉토리>₩.metadata ₩.plugins₩org.eclipse.wst.server.core₩tmp0₩work₩Catalina ₩localhost₩<프로젝트 이름> ₩org₩apache₩jsp₩helloToday_jsp.java가 존재함

# 정리

- **웹 서비스 처리 과정에서 정적인 페이지와 동적인 페이지 처리 시 차이를 살펴보았다.**
- Servlet의 정의, 등장 배경, 특징에 대하여 알아보았다.
- IDE Servlet을 생성해보고 Servlet의 구조와 생명주기에 대하여 알아보았다.
- **Servlet을 이용한 웹 요청 처리 과정에 대하여 알아보았고, Servlet과 JSP의 특징을 살펴보았으며, 비교해보았다.**