
1장. 서블릿

1.1. 학습목표

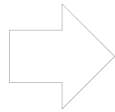
- JAVA Web 프로그램은 서블릿 → JSP → JSP MVC → Spring MVC 단계로 진행됨



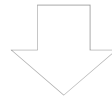
학습목표

Web 프로그램

JAVA 프로그램



JAVA
WEB 프로그램



서블릿

HTML 코드 출력이
어려움

JSP

HTML 코드 내
JSP 코드의 가독성
떨어짐

JSP MVC
(Model – 1)

Controller와 View가
같은 파일에 존재
. Controller : JAVA 코드
. View : HTML 코드

JSP MVC
(Model – 2)

Frame Work의
필요성 발생

SPRING
MVC

FrameWork의
완성

* JSP : Java Server Page

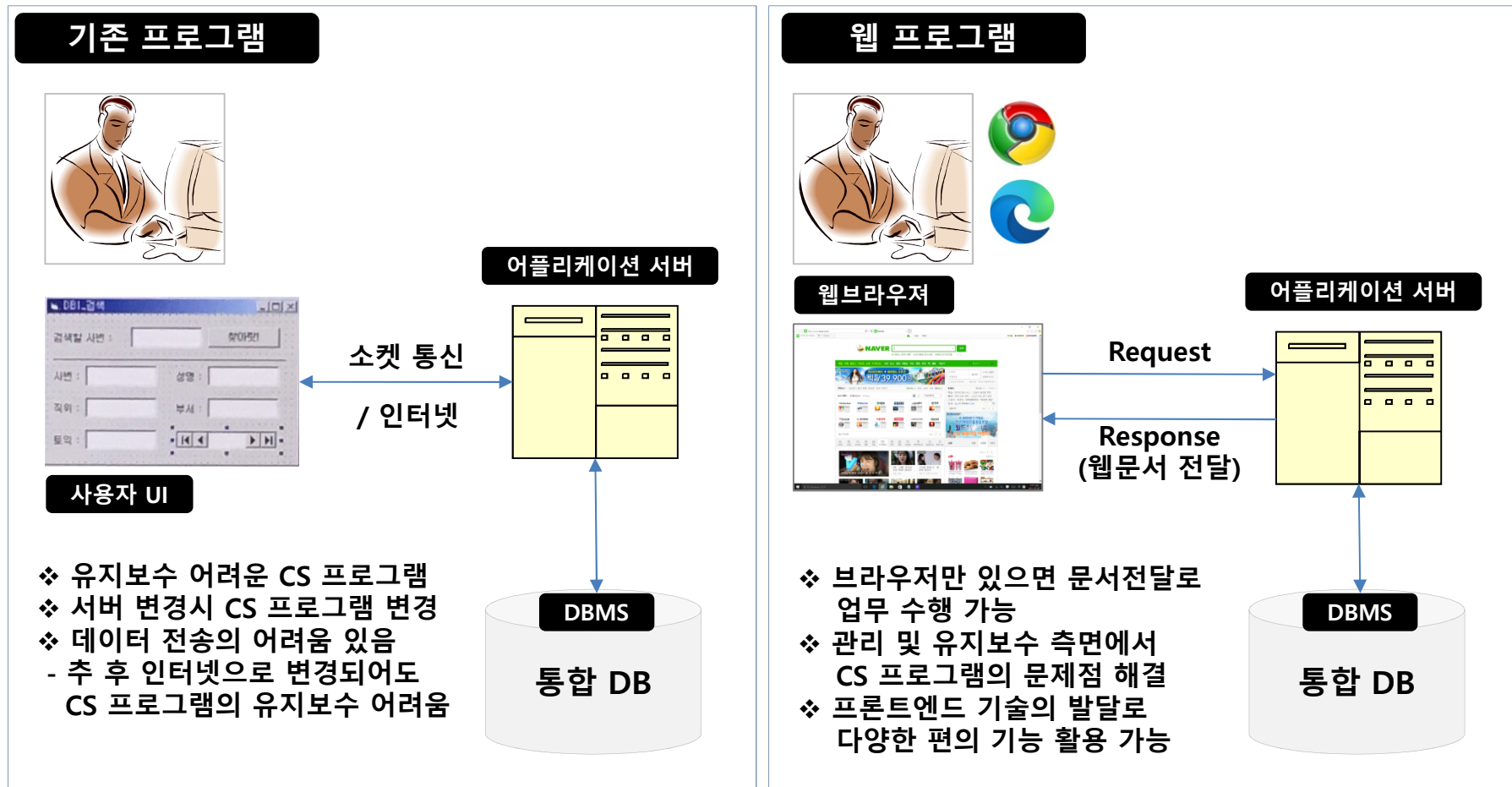
* MVC : Model – View - Controller

1.2. 웹 프로그램의 필요성

- 일반적인 업무용 프로그램은 사용자 UI / Application / Database 3단계로 이루어짐.
- 이를 3-tire 구조라고 이야기하며, 관리 및 유지보수 측면에서 유용하기 때문에 Web을 활용



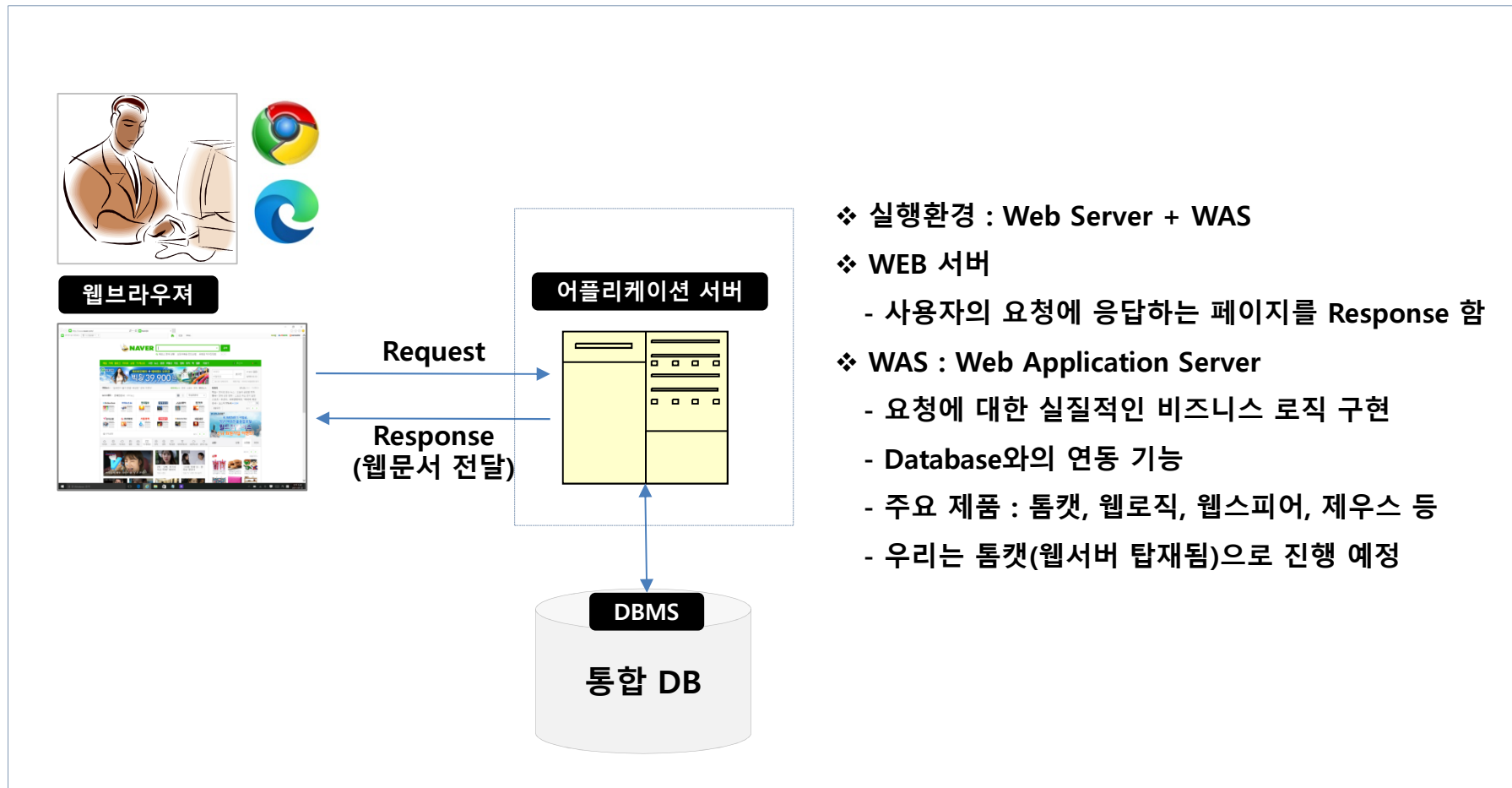
웹프로그램의 필요성



1.3. 웹 프로그램의 동작환경

- 사용자의 요청에 서버가 응답하는 환경으로 WEB의 실행환경이 필요함.
- 서버의 Application은 사용자의 요청에 실질적인 구현을 하는 업무용 프로그램임.

웹프로그램 동작 환경



1.4. 개발환경 구축

- 웹프로그램은 JAVA를 컴파일하기 위한 JDK와 Web용 Application 서버인 Tomcat 설치
- 개발환경 : JDK, Tomcat, Eclipse



개발환경 구축

JDK 설치

1. JAVA 컴파일을 위한 JDK 설치.
 - oracle 사이트에서 JDK 다운로드.

톰캣

2. 웹서버 및 WAS의 용도인 Apache Tomcat 설치
 - 아파치 톰캣 사이트에서 다운로드
 - 서비스 용도가 아닌 개발용도이므로 ZIP 파일 기반으로 설치.
인스톨러용으로 설치시 서비스가 매번 자동 실행됨.
 - 톰캣 압축 해제 이후 bin 폴더내의 startup.bat 파일 실행하여 실행되었음을 확인.
만약 에러발생시 JAVA_HOME에 Tomcat Path 설정 필요. (톰캣은 JDK를 통해 실행됨)

JAVA_HOME D:\tools\jdk-12.0.1

이클립스설치

1. Java 개발도구.
2. www.eclipse.org에서 무료 사용가능
3. J2EE 기반으로 설치.
(Eclipse IDE for Java EE Developer)
4. 서블릿은 NEW > Project > WEB > Dynamic Web Project로 프로젝트 파일 생성
만약 이클립스 실행이 안될 경우 eclipse.ini 의 Xmx의 숫자를 줄임. (256m)

2.1. 첫 서블릿 코드 생성하기

- 웹프로그램은 JAVA를 컴파일하기 위한 JDK와 Web용 Application 서버인 Tomcat 설치
- 개발환경 : JDK, Tomcat, Eclipse



첫 서블릿 코드 생성

첫 서블릿 코드

```
public class Hello extends HttpServlet
{
    public void service (HttpServletRequest request,
        HttpServletResponse response) throws IOException, ServletException
    {
        System.out.println("Hello Servlet!!");
        OutputStream os = response.getOutputStream();
        PrintStream out = new PrintStream(os, true);
        out.println("HELLO~~~ SERVLET~~~");
    }
}
```

- ❖ 서블릿은 service란 이름으로 Main을 생성함
- ❖ HttpServlet이란 추상클래스를 상속받음
- ❖ service 메서드를 호출하여 Http 프로토콜 활용

컴파일

```
C:\>javac -cp E:\JAVA_STUDY\apache-tomcat-9.0.64\lib\servlet-api.jar hello.java
```

- ❖ Servlet을 컴파일하는 도구는 JDK에 없고, Tomcat Library에 있음
- ❖ 이를 사용하기 위해서는 -cp의 옵션을 활용함.

실행

- ❖ 생성된 class 파일을 설치된(압축해제한) 톰캣 폴더의 %webapps%ROOT%WEB-INF%classes의 위치로 옮김 (classes 폴더 생성 필요)
- ❖ http://localhost:8080/hello 요청 (요청 이전에 Tomcat이 실행되어 있어야 함.)
- ❖ 기존의 Port가 있어서 실행이 안될 경우
 - netstat -a -o 로 8080 포트 확인 → taskkill /f /pid 포트번호로 8080 포트 Kill 이후 실행
 - 수정시 컴파일 → 톰캣 재실행의 과정을 거쳐야 함. (불편함. 그래서 이클립스 활용함)

2.2. 실행 파일 Mapping 방법

- http://localhost:8080/hello 요청시 2단계를 거치면서 요청에 대한 실행 파일을 찾음
- 1단계 (예약된 classes 폴더에서 찾고) → 2단계 (Tomcat의 web.xml 에 설정된 mapping을 찾음)



실행파일 단계별 Search

1단계 : 예약된 위치

내 PC > 새 볼륨 (E:) > JAVA_STUDY > apache-tomcat-9.0.64 > webapps > ROOT > WEB-INF >			
이름	수정한 날짜	유형	크기
classes	2022-06-30 오후 8:23	파일 폴더	
web.xml	2022-06-30 오후 8:15	XML 문서	2KB

내 PC > 새 볼륨 (E:) > JAVA_STUDY > apache-tomcat-9.0.64 > webapps > ROOT > WEB-INF > classes >			
이름	수정한 날짜	유형	크기
Hello.class	2022-06-30 오후 8:22	CLASS 파일	1KB

❖ Web Browser에서

- http://localhost:8080/hello 요청시 위의 class파일 대상으로 실행함.
- ❖ classes 폴더는 예약된 위치로써 서블릿이 컴파일된 클래스 파일을 실행하는 영역임.
- ❖ 참고 :
 - WEB-INF 폴더는 서버만 호출할 수 있는 것이며 사용자가 직접적으로 호출 불가함.

2단계 : web.xml mapping

내 PC > 새 볼륨 (E:) > JAVA_STUDY > apache-tomcat-9.0.64 > webapps > ROOT > WEB-INF >			
이름	수정한 날짜	유형	크기
classes	2022-06-30 오후 8:23	파일 폴더	
web.xml	2022-06-30 오후 8:15	XML 문서	2KB

```
<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.jcp.org/xml/ns/javaee/web-app_3_2.xsd"
  version="4.0"
  metadata-complete="true">
  <servlet>
    <servlet-name>Hi</servlet-name>
    <servlet-class>Hello</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Hi</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>
```

name : mapping 이름
class : 실행 클래스

http://localhost:8080/hello

- ❖ 1단계에서 파일이 없을 경우 web.xml의 mapping을 대상으로 WAS에서 회신함.

2.3. 서블릿에서의 출력

- 서블릿은 HTML의 문서의 형태로 Web Browser에서 출력하는 것을 목적으로 함
- 서블릿의 출력방식 : HTML 문서, Console (톰캣 실행파일)



서블릿에서의 출력

console에 출력

```
public class Hello extends HttpServlet
{
    public void service (HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException, ServletException
    {
        System.out.println("Hello Sevlet!!");

        OutputStream os = response.getOutputStream();
        PrintStream out = new PrintStream(os, true);
        out.println("HELLO~~~ SERVLET~~~");
    }
}
```

- ❖ System.out.println을 통한 console에 출력
- ❖ 주로 log를 확인할 때 사용함.

HTML 문서로 출력

```
public class Hello extends HttpServlet
{
    public void service (HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException, ServletException
    {
        OutputStream os = response.getOutputStream();
        PrintStream out = new PrintStream(os, true);
        out.println("HELLO~~~ SERVLET~~~");
    }
}
```

- ❖ HttpServletRequest는 입력도구
- ❖ HttpServletResponse는 출력도구
- ❖ response.getOutputStream으로 네트워크 스트림 생성
- ❖ print를 위해서 PrintStream으로 객체를 재생성
- ❖ buffer를 활용하기 위해 true 옵션 활용 (flush)
- ❖ PrintStream을 제공하는 Getter활용

```
PrintWriter out = response.getWriter();
out.println("HELLO~~~ SERVLET~~~");
```

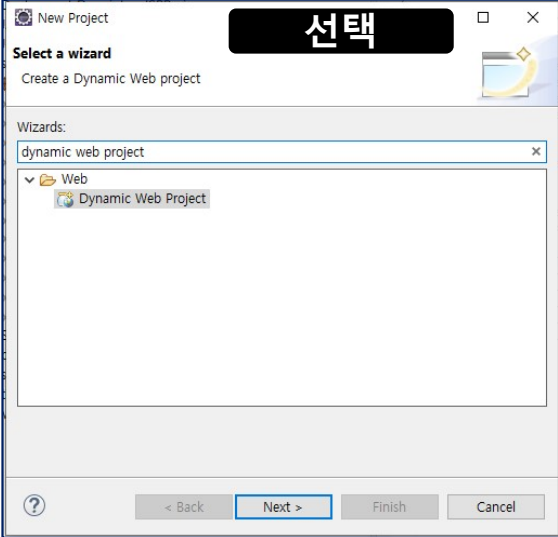
* 참고 : JAVA에서는 다국어 계열은 Stream이 아닌 Writer를 사용함

2.4. eclipse의 활용

- 소스가 수정될 때마다 컴파일, class파일 이동, WAS 재실행등의 불편함이 있어 IDE를 활용
- 프로젝트는 Dynamic Web Project로 시작하며, Tomcat을 설정해야 사용가능함.

Eclipse로 처음 시작하기

프로젝트 생성



선택

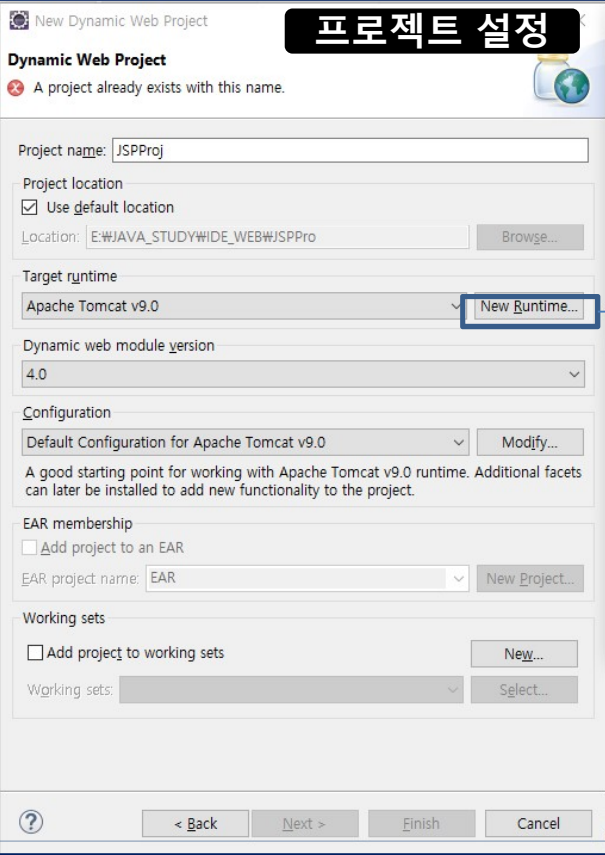
New Project
Select a wizard
Create a Dynamic Web project

Wizards:
dynamic web project

Web
Dynamic Web Project

< Back Next > Finish Cancel

프로젝트 설정



프로젝트 설정

New Dynamic Web Project

A project already exists with this name.

Project name: JSPProj

Project location
☒ Use default location
Location: E:\JAVA_STUDY\WIDE_WEB\JSPPro Browse...

Target runtime
Apache Tomcat v9.0 New Runtime...

Dynamic web module version
4.0

Configuration
Default Configuration for Apache Tomcat v9.0 Modify...

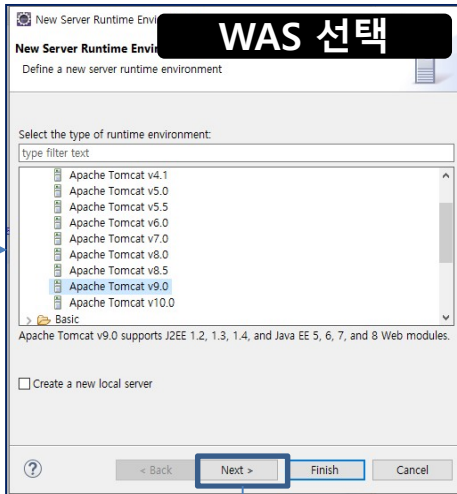
A good starting point for working with Apache Tomcat v9.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership
☐ Add project to an EAR
EAR project name: EAR New Project...

Working sets
☐ Add project to working sets New...
Working sets: Select...

< Back Next > Finish Cancel

WAS 선택



WAS 선택

New Server Runtime Environment

Define a new server runtime environment

Select the type of runtime environment:
type filter text

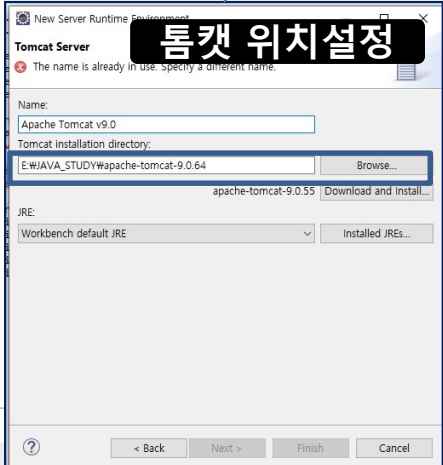
- Apache Tomcat v4.1
- Apache Tomcat v5.0
- Apache Tomcat v5.5
- Apache Tomcat v6.0
- Apache Tomcat v7.0
- Apache Tomcat v8.0
- Apache Tomcat v8.5
- Apache Tomcat v9.0
- Apache Tomcat v10.0

Basic
Apache Tomcat v9.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, 7, and 8 Web modules.

☐ Create a new local server

< Back Next > Finish Cancel

톰캣 위치설정



톰캣 위치설정

New Server Runtime Environment

Tomcat Server

The name is already in use. Specify a different name.

Name: Apache Tomcat v9.0

Tomcat installation directory: E:\JAVA_STUDY\apache-tomcat-9.0.64 Browse...
apache-tomcat-9.0.55 Download and Install...

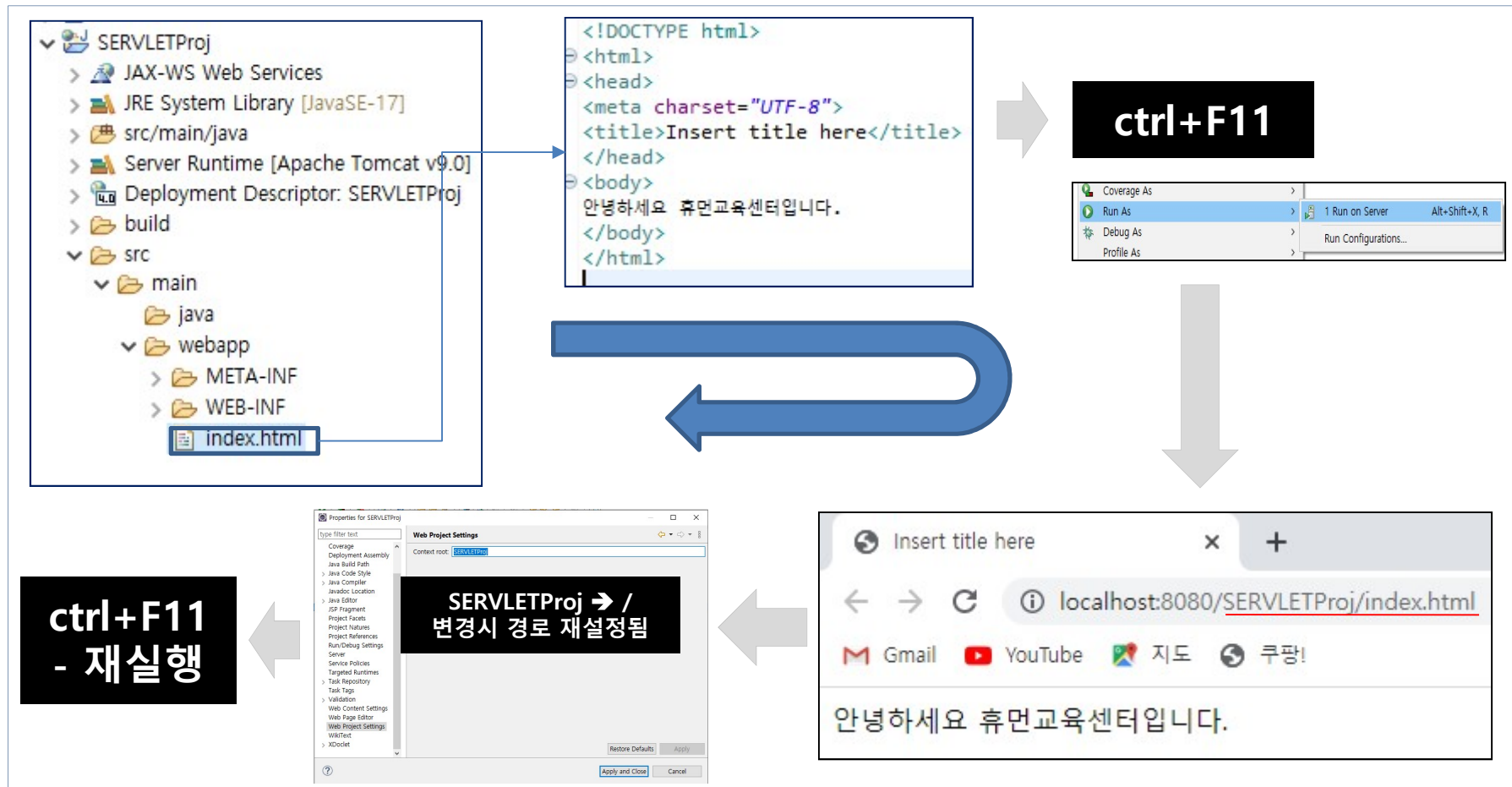
JRE: Workbench default JRE Installed JREs...

< Back Next > Finish Cancel

2.5. eclipse의 활용 첫 웹프로그램

- Eclipse에서는 WAS(톰캣)이 연결되어 있어 편리하게 사용가능함.
- 요청의 root를 설정하기 위해서는 property → Web Project Setting에서 처리 가능함.

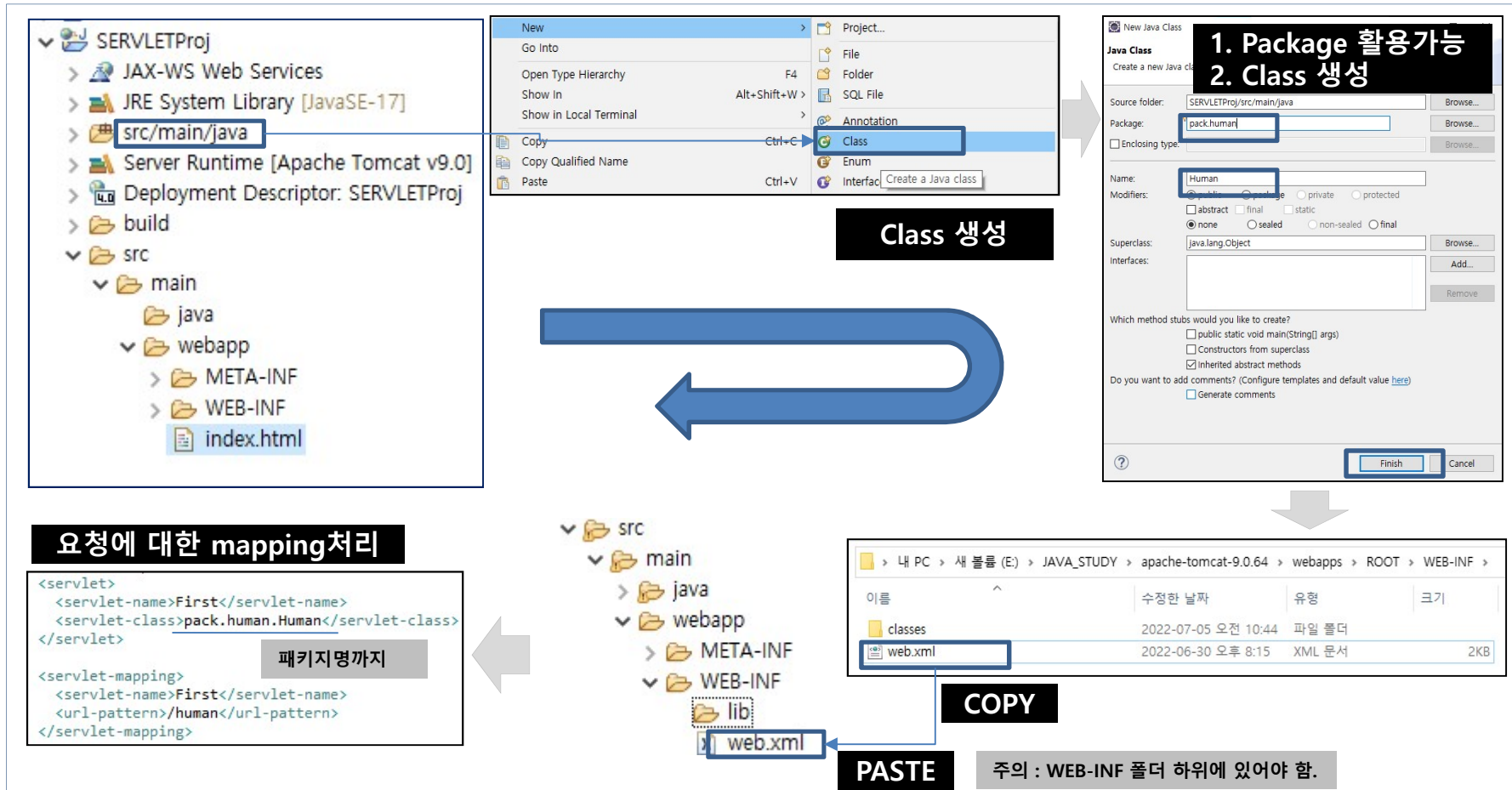
Eclipse로 처음 시작하기 (HTML)



2.5.1. eclipse의 활용 첫 서블릿(1)

- 서블릿은 src/main/java 에서 class를 만들어 사용함.
- 요청에 대한 class 명을 mapping하기 위해 web.xml을 활용함. (web.xml 카피하여 사용할 것)

Eclipse로 처음 시작하기 (Servlet)



1. Package 활용가능
2. Class 생성

Class 생성

요청에 대한 mapping처리

```
<servlet>
  <servlet-name>First</servlet-name>
  <servlet-class>pack.human.Human</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>First</servlet-name>
  <url-pattern>/human</url-pattern>
</servlet-mapping>
```

패키지명까지

COPY

PASTE

주의 : WEB-INF 폴더 하위에 있어야 함.

2.5.2. eclipse의 활용 첫 서블릿(2)

- 서블릿의 요청은 WEB.XML에서 URL-Pattern에 맞추어 요청해야 함.

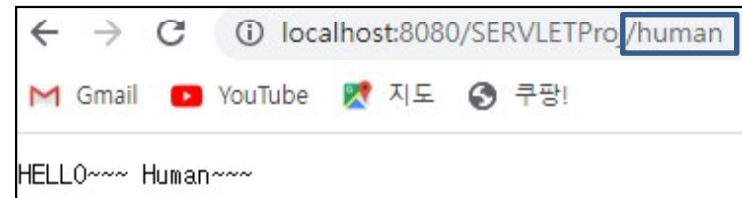


Eclipse로 처음 시작하기 (Servlet)

Servlet 코드

```
public class Human extends HttpServlet{  
    public void service (HttpServletRequest request,  
                        HttpServletResponse response)  
        throws IOException, ServletException  
    {  
        PrintWriter out = response.getWriter();  
        out.println("HELLO~~~ Human~~~");  
    }  
}
```

ctrl+F11



❖ 참고사항.

- Context Root 가 현재는 SERVLETPro이나
Context Root 를 '/'로 바꿀 경우 요청의 경로는 바뀌어짐
- Context Root 는 Property → Web Project Setting에서 설정 가능

2.6. Annotation 활용

- WEB.XML 에서 늘 mapping 처리가 어려워서 annotation으로 처리함.
- Class 명 위에 @WebServlet("/human") 으로 설정함.



Annotation 활용한 요청 처리

ANNOTATION 처리

```
@WebServlet("/human")
public class Human extends HttpServlet {
    public void service (HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {
        {
            PrintWriter out = response.getWriter();
            out.println("HELLO~~~ Human~~~");
        }
    }
}
```

Human.java

WEB.XML의 내용 변경

```
<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.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
    version="4.0"
    metadata-complete="false">
    <servlet>
        <servlet-name>First</servlet-name>
        <servlet-class>pack.human.Human</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>First</servlet-name>
        <url-pattern>/human</url-pattern>
    </servlet-mapping>
</web-app>
```

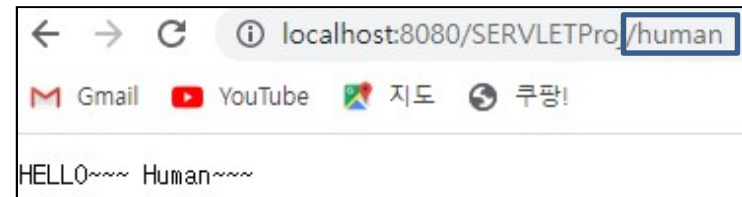
변경 : true → false

삭제

ctrl+F11

❖ 참고사항.

- Annotation 사용시 Servlet 코드의 Class 명은 요청의 처리와 상관이 없음.
- @WebServlet('/hi') 로 변경시 요청이 hi로 들어와야만 처리됨.
- 여러 사람이 동시에 개발할 때 공용 파일인 WEB.XML의 파일을 처리안해도 되는 장점이 있음



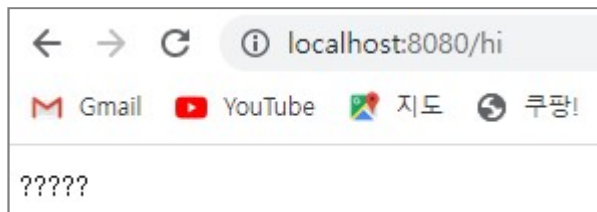
2.7. 한글 처리 문제

- 다국어 처리에 대해서는 늘 UTF-8로 설정하여 사용하는 것이 좋음.
- Response는 출력시, Request는 요청시의 문자 인코딩 방법임.

한글 처리에 대한 Setting

한글깨짐 현상

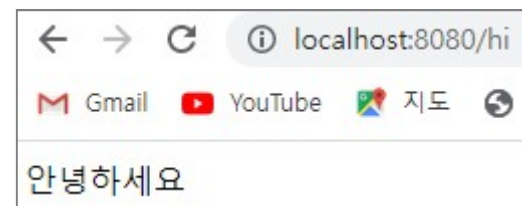
```
@WebServlet("/hi")
public class Human extends HttpServlet{
    public void service (HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException, ServletException
    {
        PrintWriter out = response.getWriter();
        out.println("안녕하세요");
    }
}
```



한글깨짐 보완

```
@WebServlet("/hi")
public class Human extends HttpServlet{
    public void service (HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException, ServletException
    {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        PrintWriter out = response.getWriter();
        out.println("안녕하세요");
    }
}
```




2.8.1. GET 요청

- Get 요청에 대한 Servlet 실습

GET 요청

GET 요청

웹브라우저



어플리케이션 서버



Request

Response

- ❖ Request 시 쿼리스트링을 통해 요청
 - localhost:8080 / hi
 - localhost:8080 / hi ? cnt=3
- ❖ Response 시 쿼리스트링을 감안하여 처리한 웹문서 전달

GET 요청 실습

```
@WebServlet("/hi")
public class Human extends HttpServlet{
    public void service (HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException, ServletException
    {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        PrintWriter out = response.getWriter();

        int intCnt = 0;
        String strCnt = request.getParameter("cnt");

        // strCnt가 NULL이거나 빈문자열일 때는 intCnt=0으로 설정
        if ( strCnt!=null && !strCnt.equals("") ) {
            intCnt = Integer.parseInt(strCnt);
        }
        for (int i=0; i<intCnt; i++) {
            out.println("<H1>HI!! HUMAN</H1>");
        }
    }
}
```

Human.java

← → ↻ ⓘ localhost:8080/hi?cnt=3

Gmail YouTube 지도 쿠팡!

2.8.2. 입력폼을 통한 GET 요청

- 입력폼을 통한 GET 요청 실습
- 페이지 단위로 값을 전달하는 속성은 name임.

GET 요청

입력 폼

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action="/hi" method="get">
  <input type="text" value="0" name="cnt"/>
  <input type="submit" value="클릭" />
</form>
</body>
</html>
```

hello-input.html

localhost:8080/hello-input.html

Gmail YouTube 지도 쿠팡!

2

❖ UTF-8 설정 방법

- Window > Preference > WEB 에서 설정

Servlet 코드 : 앞페이지 코드

Name에 해당하는 것이 페이지간 연결고리
(단, 문자로 연결됨)

```
int intCnt = 0;
String strCnt = request.getParameter("cnt");

// strCnt가 NULL이거나 빈문자열일 때는 intCnt=0으로 설정
if ( strCnt!=null && !strCnt.equals("") ) {
    intCnt = Integer.parseInt(strCnt);
}
for (int i=0; i<intCnt; i++) {
    out.println("<H1>HI!! HUMAN</H1>");
}
```

클릭시 get
요청

localhost:8080/hi?cnt=5

Gmail YouTube 지도 쿠팡!

HI!! HUMAN
HI!! HUMAN
HI!! HUMAN
HI!! HUMAN
HI!! HUMAN

2.8.3. 입력폼을 통한 POST 요청

- POST 전달 방식은 전달할 내용이 많을 때 Post 방식으로 처리함.
- GET은 쿼리스트링 방식이나 POST는 Body안에 form data로 전달함.

POST요청

입력 폼

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action="/Calc-Post" method="post">
  <input type="text" value="0" name="x">
  <input type="text" value="0" name="y">
  <input type="submit" value="더하기" />
</form>
</body>
</html>
```

Calc-post.html

localhost:8080/calc-post.html

Gmail YouTube 지도 쿠팡!

3 4 더하기

Servlet 코드

CalcPostTest.java

```
@WebServlet("/Calc-Post")
public class CalcPostTest extends HttpServlet{
    public void service (HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        String strX = request.getParameter("x");
        String strY = request.getParameter("y");

        int intX = 0;
        int intY = 0;

        if (strX != null && !strX.equals("")) {
            intX = Integer.parseInt(strX);
        }
        if (strY != null && !strY.equals("")) {
            intY = Integer.parseInt(strY);
        }

        int result = intX + intY;

        PrintWriter out = response.getWriter();
        out.println ("두수의 합은 = " + result);

    }
}
```

클릭시 post
요청

2.8.4. 계산기 실습(1)

- 계산기 실습



계산기 실습

입력 폼

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<style>
    .box {
        width : 50px ;
        text-align : right
    }
    .button {
        width : 50px;
        text-align : justify
    }
</style>
</head>
<body>
    <form action="/Calculator" method="post">
        <div>
            <input class="box" type="text" value="0" name="x"/>
            <input class="box" type="text" value="0" name="y"/>
        </div>
        <div>
            <input class="button" type="submit" value="더하기" name="operator"/>
            <input class="button" type="submit" value="빼기" name="operator"/>
        </div>
    </form>
</body>
</html>
```

calculator.html

Servlet 코드

Calculator.java

```
@WebServlet ("/Calculator")
public class Calculator extends HttpServlet {
    public void service (HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        int result = 0;
        int intX = Integer.parseInt(request.getParameter("x"));
        int intY = Integer.parseInt(request.getParameter("y"));
        String operator = request.getParameter("operator");

        if (operator.equals("더하기")) result = intX + intY;
        else if (operator.equals("빼기")) result = intX - intY;

        PrintWriter out = response.getWriter();
        out.printf ("결과는 %d입니다. ", result);
    }
}
```

- ❖ 입력폼을 통해서는 null 또는 빈문자열이 없을 것이기 때문에 이를 판단하는 조건문은 코드상 없음.
실전에서는 꼭 있는 것이 좋음

2.8.5. 계산기 실습(2)

- 배열을 활용한 계산기 실습



계산기 실습 (배열 활용)

입력 폼

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<style>
    .box {
        width : 50px ;
        text-align : right
    }
    .button {
        width : 50px;
        text-align : justify
    }
</style>
</head>
<body>
    <form action="/Calculator2" method="post">
        <div>
            <input class="box" type="text" value="0" name="values"/>
            <input class="box" type="text" value="0" name="values"/>
        </div>
        <div>
            <input class="button" type="submit" value="더하기" name="operator"/>
            <input class="button" type="submit" value="빼기" name="operator"/>
        </div>
    </form>
</body>
</html>
```

calculator2.html

같은 변수로 넘어갈 경우
배열로 처리 가능

Servlet 코드

Calculator.java

```
@WebServlet ("/Calculator2")
public class Calculator2 extends HttpServlet{
    public void service (HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        int result = 0;
        String[] strV = request.getParameterValues("values");
        String operator = request.getParameter( operator );

        int intX = Integer.parseInt(strV[0]);
        int intY = Integer.parseInt(strV[1]);

        if (operator.equals("더하기")) result = intX + intY;
        else if (operator.equals("빼기")) result = intX - intY;

        PrintWriter out = response.getWriter();
        out.printf ("결과는 %d입니다. ", result);

    }
}
```

2.9.1. 상태유지 방법 (페이지간 데이터 공유방법)

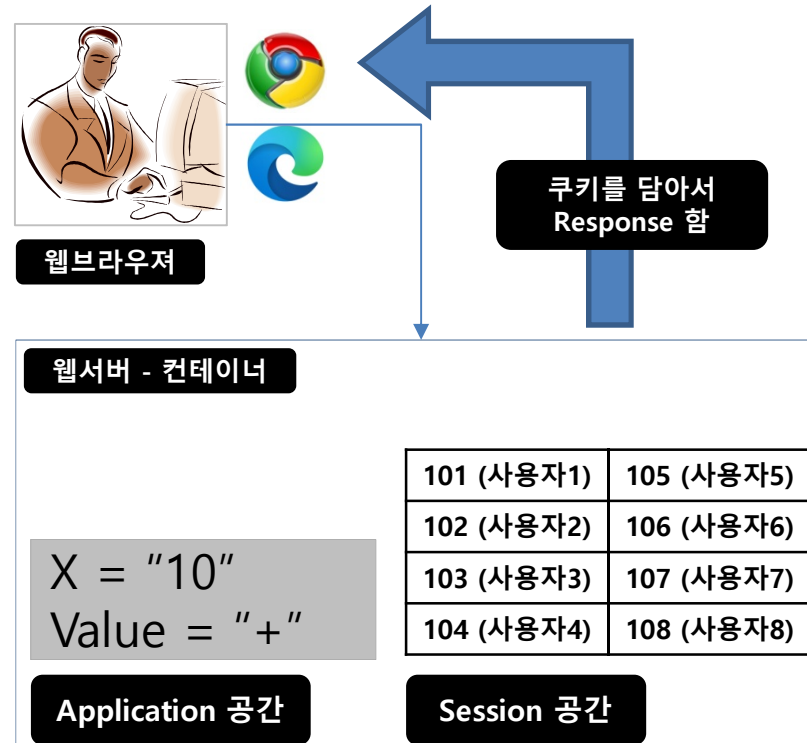
- 객체 활용 : Application, Session, Cookie
- 그 외 : hidden input , query string (get 방식에서 사용)

상태유지 방법

상태유지 방법 - 객체

- ❖ Application
 - Application (웹프로그램, 서블릿)이 동작하고 있는 동안 모든 접속에 대해서 동작함.
 - 모든 사용자가 같은 데이터를 공유함.
- ❖ Session
 - Session은 동일 접속 기준으로만 동작함.
 - Web Browser가 닫히면 session_id는 사라짐
 - 접속된 사용자별로 다른 정보 공유됨
- ❖ Cookie
 - Cookie는 클라이언트 단에 저장되는 사용자 정보

객체별 동작방식



2.9.2. 상태유지 방법 (Application 사용)

- Application 객체는 모든 사용자가 같은 데이터를 사용하는 것.
- WAS가 실행되는 동안 같은 데이터로 처리됨.



상태유지 방법 (Application)

입력 폼

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<style>
    .box {
        width : 50px ;
        text-align : right
    }
    .button {
        width : 50px;
        text-align : justify
    }
</style>
</head>
<body>
    <form action="/CalculatorApplication" method="post">
        <div>
            <input class="box" type="text" value="0" name="value"/>
        </div>
        <div>
            <input class="button" type="submit" value="+" name="operator"/>
            <input class="button" type="submit" value="-" name="operator"/>
            <input class="button" type="submit" value="=" name="operator"/>
        </div>
    </form>
</body>
</html>
```

Calculator-application.html

Application 활용

```
@WebServlet ("/CalculatorApplication")
public class CalcApplication extends HttpServlet{
    public void service (HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        // application Context 얻음
        ServletContext app = request.getServletContext();
        PrintWriter out = response.getWriter();

        int result = 0;
        String strValue = request.getParameter("value");
        String operator = request.getParameter("operator");

        // +, - 일 때는 application 객체에 x라는 변수로 담음
        if (operator.equals("+") || operator.equals("-")) {
            app.setAttribute("x", strValue);
            app.setAttribute("oper", operator);
        }
        else if (operator.equals("=")) {
            int intValue = Integer.parseInt(strValue);
            if (app.getAttribute("oper").equals("+")) {
                result = Integer.parseInt((String) app.getAttribute("x")) + intValue;
            }
            else if (app.getAttribute("oper").equals("-")) {
                result = Integer.parseInt((String) app.getAttribute("x")) - intValue;
            }
            out.printf ("결과는 %d입니다. ", result);
        }
    }
}
```

CalcApplication.java

Application에 저장

Application에서 조회

2.9.3. 상태유지 방법 (Session사용)

- Session 객체는 사용자별로 구분하여 처리됨.
- Session 은 Browser를 닫으면 사라짐



상태유지 방법 (Session)

입력 폼

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<style>
    .box {
        width : 50px ;
        text-align : right
    }
    .button {
        width : 50px;
        text-align : justify
    }
</style>
</head>
<body>
<form action="/CalculatorSession" method="post">
    <div>
        <input class="box" type="text" value="0" name="value"/>
    </div>
    <div>
        <input class="button" type="submit" value="+" name="operator"/>
        <input class="button" type="submit" value="-" name="operator"/>
        <input class="button" type="submit" value="=" name="operator"/>
    </div>
</form>
</body>
</html>
```

Calculator-session.html

Session 활용

```
@WebServlet ("/CalculatorApplication")
public class CalcApplication extends HttpServlet{
    public void service (HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        // application Context 얻음
        ServletContext app = request.getServletContext();
        PrintWriter out = response.getWriter();

        int result = 0;
        String strValue = request.getParameter("value");
        String operator = request.getParameter("operator");

        // +, - 일 때는 application 객체에 x라는 변수로 담음
        if (operator.equals("+") || operator.equals("-")) {
            app.setAttribute("x", strValue);
            app.setAttribute("oper", operator);
        }
        else if (operator.equals("=")) {
            int intValue = Integer.parseInt(strValue);
            if (app.getAttribute("oper").equals("+")) {
                result = Integer.parseInt((String) app.getAttribute("x")) + intValue;
            }
            else if (app.getAttribute("oper").equals("-")) {
                result = Integer.parseInt((String) app.getAttribute("x")) - intValue;
            }
            out.printf ("결과는 %d입니다. ", result);
        }
    }
}
```

CalcSession.java

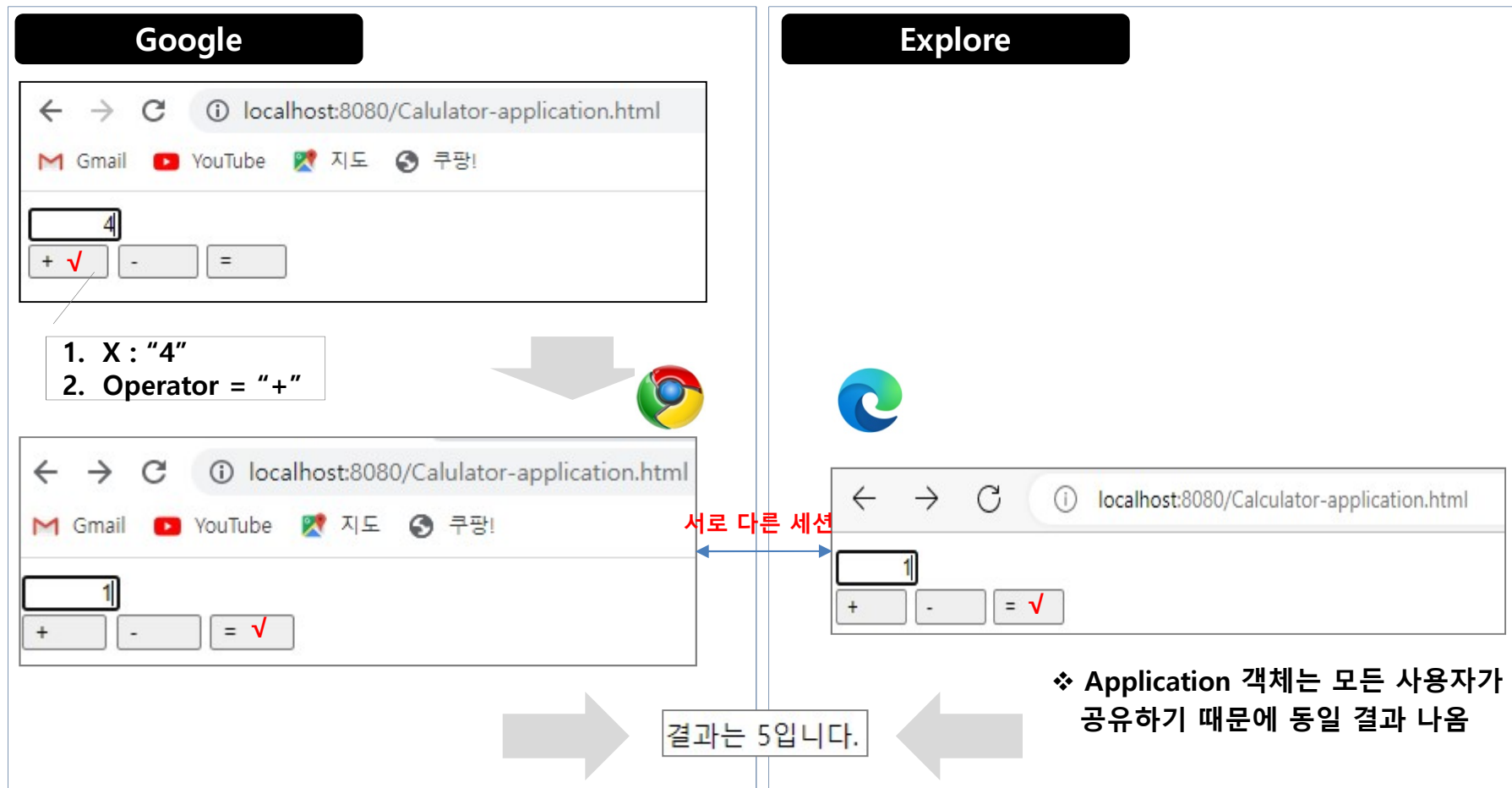
Session에 저장

Session 에서 조회

2.9.4. Application과 Session의 차이 분석 - Application

- Session은 브라우저를 닫으면 사라지지만, Application은 유지됨.
- Session은 브라우저가 다르면 다르게 적용되지만, Application은 동일하게 유지됨.

상태유지 방법 (Application)

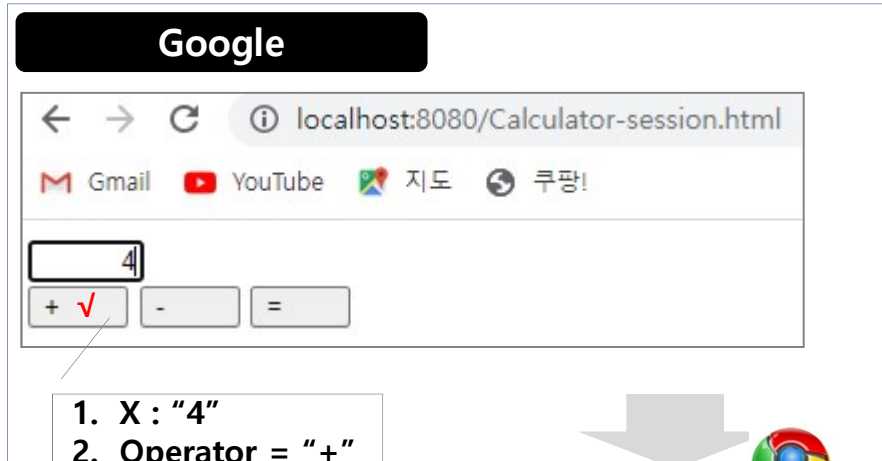


2.9.5. Application과 Session의 차이 분석 - Session

- Session은 브라우저를 닫으면 사라지지만, Application은 유지됨.
- Session은 브라우저가 다르면 다르게 적용되지만, Application은 동일하게 유지됨.

상태유지 방법 (Session)

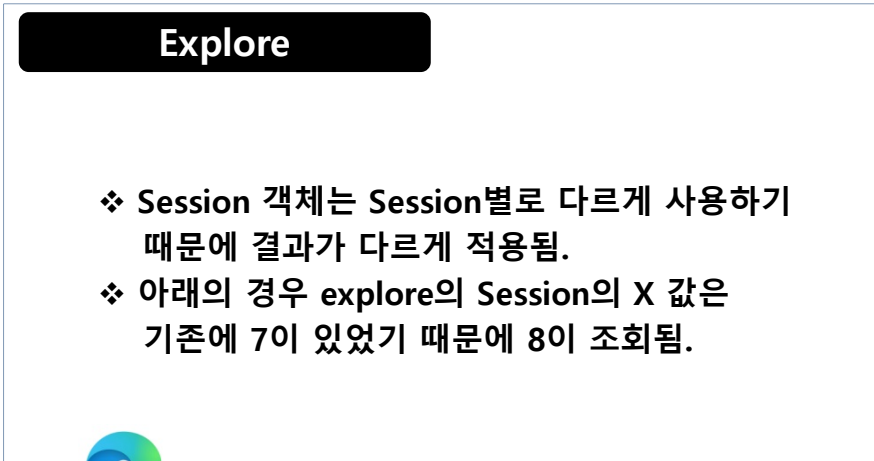
Google



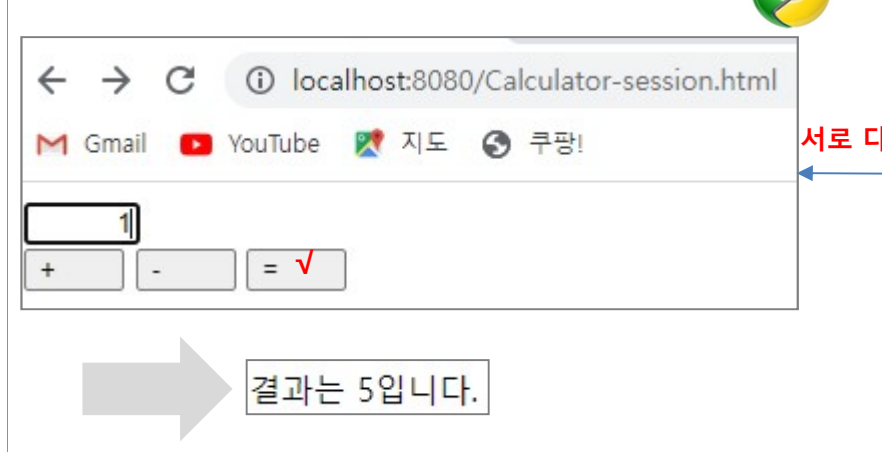
1. X : "4"
2. Operator = "+"

Explore

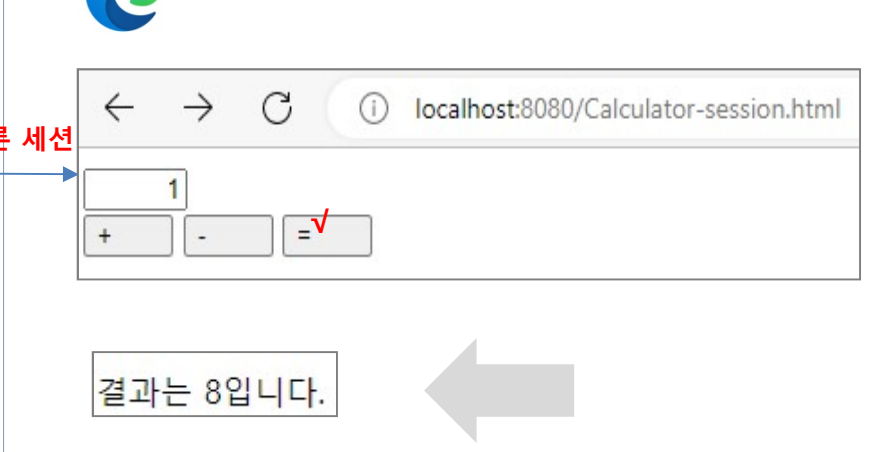
❖ Session 객체는 Session별로 다르게 사용하기 때문에 결과가 다르게 적용됨.
❖ 아래의 경우 explore의 Session의 X 값은 기존에 7이 있었기 때문에 8이 조회됨.



서로 다른 세션



결과 5입니다.



결과 8입니다.

2.9.6. Application과 Session의 관련 메서드

- 내장객체인 Application / Session 관련 메서드에 대해서 설명



내장객체 관련 메서드

Application

메서드	설명
setAttribute(string, value)	Attribute 등록
getAttribute(string)	Attribute 조회
getServerInfo()	컨테이너의 이름과 버전 반환
getContextPath()	웹 애플리케이션의 URL 경로 중 컨텍스트 패스명을 반환
getRealPath()	JSP의 실제 경로 반환
getMimeType(filename)	지정된 파일의 MIME 타입을 반환
Log(message)	지정된 message의 로그 저장

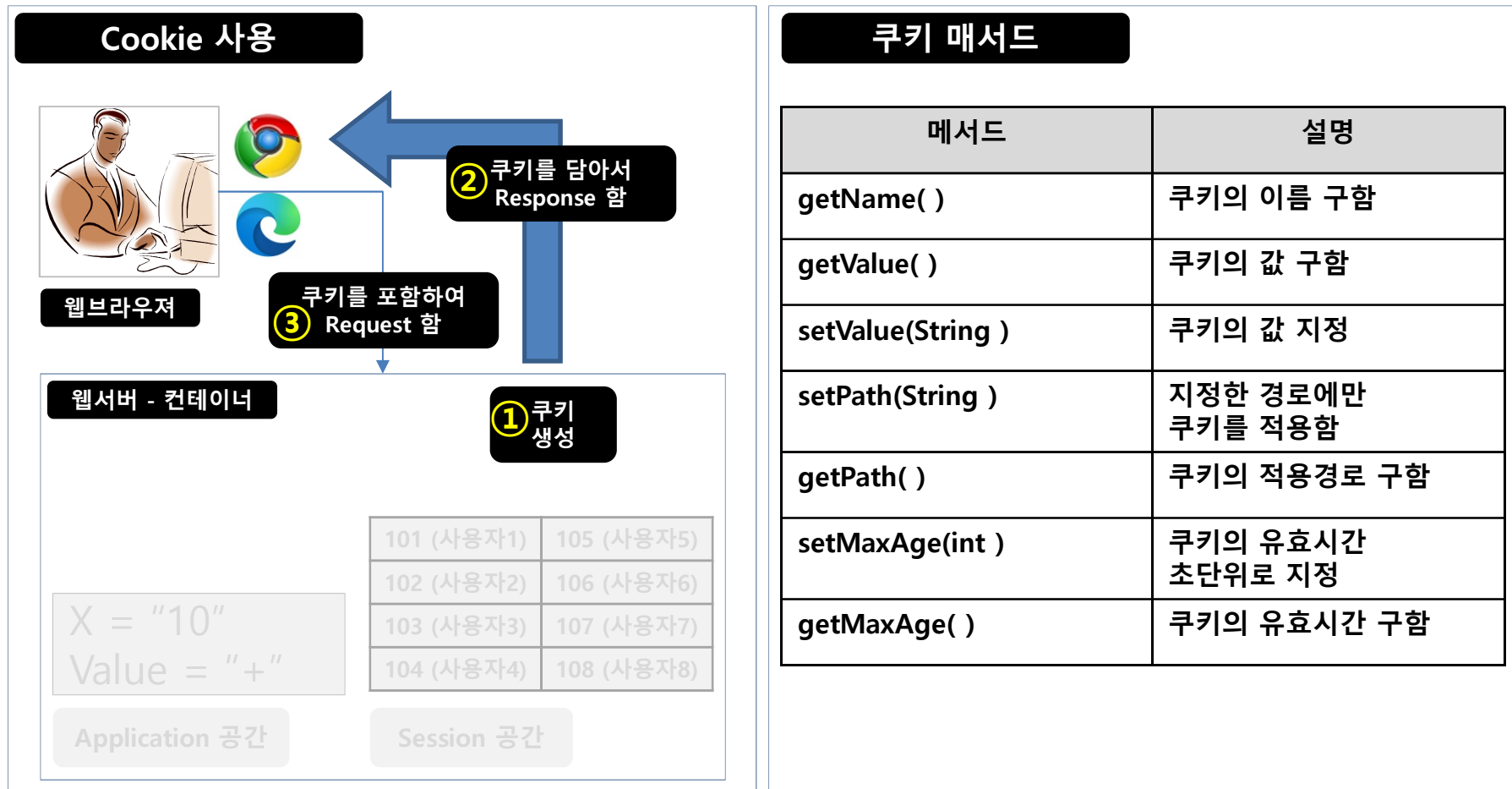
Session

메서드	설명
setAttribute(string, value)	Attribute 등록
getAttribute(string)	Attribute 조회
getAttributeNames()	세션에서 사용되는 모든 속성을 조회함.
invalidate()	세션에서 사용되는 모든 객체를 해제함
isNew()	세션이 새로 생성되었는지 확인함.
getCreationTime()	세션이 시작된 시간. - 1970.1.1 이후 밀리초
getLastAccessedTime()	마지막 요청시간 - 1970.1.1 이후 밀리초

2.9.6. Cookie 사용 방법

- Cookie는 서버가 클라이언트에 저장하는 정보로써 연결이 끊어져도 유지할 수 있음
- 요청시 header 정보에 포함되어 요청(Request)됨

Cookie 사용방법



2.9.7. Cookie 실습(1)

- Cookie 실습



Cookie 실습

입력 폼

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<style>
    .box {
        width : 50px ;
        text-align : right
    }
    .button {
        width : 50px;
        text-align : justify
    }
</style>
</head>
<body>
<form action="/CalculatorCookie" method="post">
<div>
<input class="box" type="text" value="0" name="value"/>
</div>
<div>
<input class="button" type="submit" value="+" name="operator"/>
<input class="button" type="submit" value="-" name="operator"/>
<input class="button" type="submit" value="=" name="operator"/>
</div>
</form>
</body>
</html>
```

Calculator-cookie.html

쿠키 매서드

CalcCookie.java

```
@WebServlet ("/CalculatorCookie")
public class CalcCookie extends HttpServlet{
    public void service (HttpServletRequest request, HttpServletResponse response) {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        Cookie[] cookies = request.getCookies();
        PrintWriter out = response.getWriter();

        int result = 0;
        String strValue = request.getParameter("value");
        String operator = request.getParameter("operator");

        // +, - 일 때는 cookie에 담음
        if (operator.equals("+") || operator.equals("-")) {
            // 쿠키 생성
            Cookie valueCookie = new Cookie("val", String.valueOf(strValue));
            Cookie operatorCookie = new Cookie("oper", String.valueOf(operator));
            // 생성된 쿠키를 클라이언트에 보냄.
            response.addCookie(valueCookie);
            response.addCookie(operatorCookie);
        }
        else if (operator.equals("=")) {
            int intValue = Integer.parseInt(strValue);
            int cookieVal = 0;
            String cookieOper = "";
            for (Cookie c : cookies) {
                if (c.getName().equals("val")) {
                    cookieVal = Integer.parseInt(c.getValue()); break;
                }
            }
            for (Cookie c : cookies) {
                if (c.getName().equals("oper")) {
                    cookieOper = c.getValue(); break;
                }
            }

            if (cookieOper.equals("+")) result = cookieVal + intValue;
            else if (cookieOper.equals("-")) result = cookieVal - intValue;
            out.printf ("결과는 %d입니다. ", result);
        }
    }
}
```

1. Cookie에 생성
2. Cookie 회신
→ 클라이언트에 저장

3. Cookie에 값 얻어옴

2.9.8. Cookie 실습(2)

- Cookie 메서드 실습



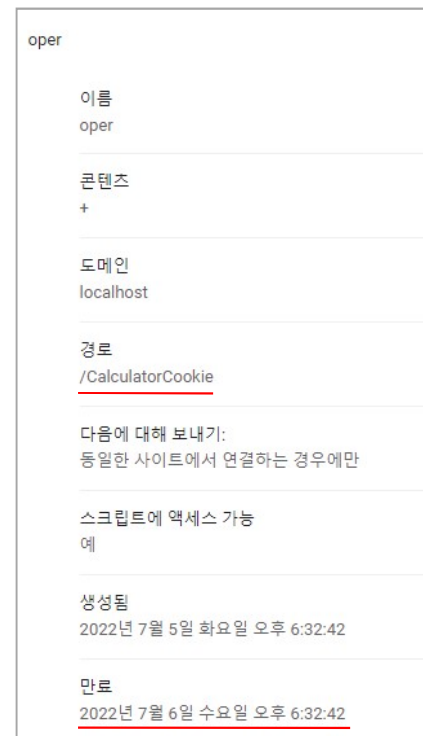
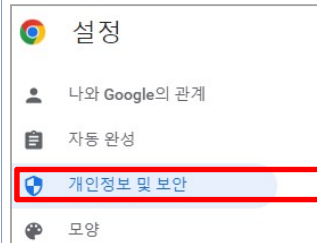
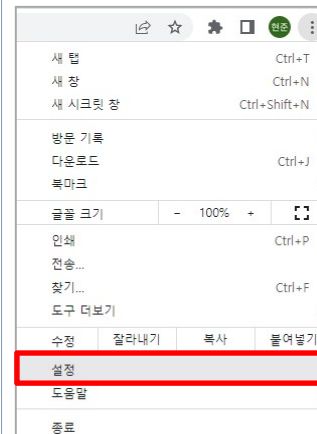
Cookie 메서드 실습

Path/ MaxAge

Calculator-path.html

```
// +, - 일 때는 cookie에 담음
if (operator.equals("+") || operator.equals("-")) {
    // 쿠키 생성
    Cookie valueCookie = new Cookie("val", String.valueOf(strValue));
    Cookie operatorCookie = new Cookie("oper", String.valueOf(operator));
    // Path 설정
    valueCookie.setPath("/CalculatorCookie");
    operatorCookie.setPath("/CalculatorCookie");
    // 쿠키 유효기간 설정 (단위 : 초. 아래의 예는 1일임)
    valueCookie.setMaxAge(24*60*60);
    operatorCookie.setMaxAge(24*60*60);
    // 생성된 쿠키를 클라이언트에 보냄.
    response.addCookie(valueCookie);
    response.addCookie(operatorCookie);
}
```

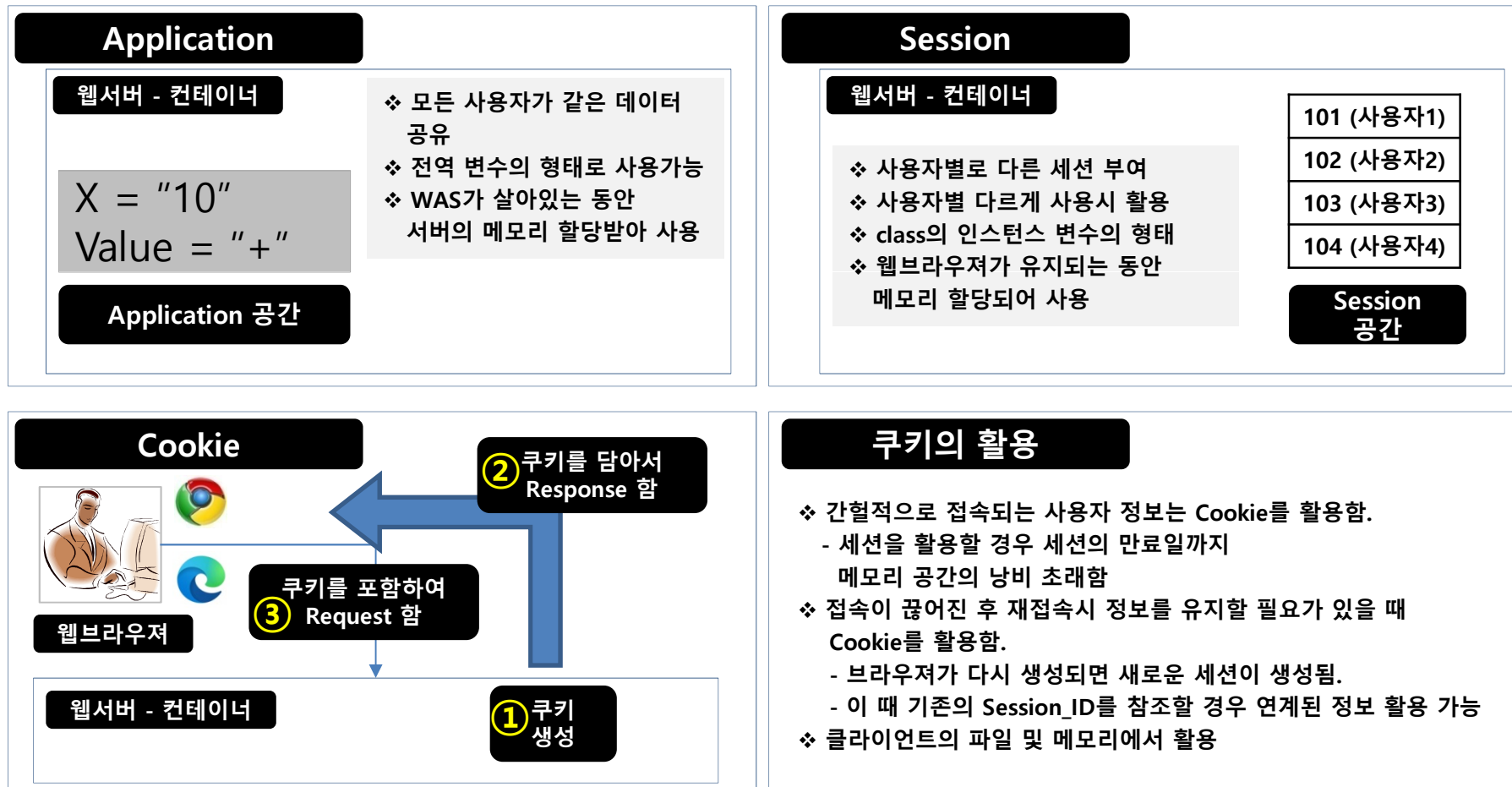
쿠키 설정 확인



2.9.9. Application / Session / Cookie 활용 방안

- Application / Session 은 서버단의 메모리를 할당받은 상태이므로, 적절하게 사용이 필요함.
- Cookie의 경우 클라이언트에서 관리하는 서버의 정보이므로 재접속 시 유용하게 사용 가능함.

Application, Session, Cookie의 활용방안



3.1. 페이지 전환

- 페이지 전환은 Redirect와 Forward가 있음.
- Redirect는 페이지 Link의 개념이고, Forward는 현재 페이지의 정보를 전달해 줌



페이지 전환

Redirect

CalcCookieRedirect.java

```
// +, - 일 때는 cookie에 담음
if (operator.equals("+") || operator.equals("-")) {
    // 쿠키 생성
    Cookie valueCookie = new Cookie("val", String.valueOf(strValue));
    Cookie operatorCookie = new Cookie("oper", String.valueOf(operator));
    // 생성된 쿠키를 클라이언트로 보냄.
    response.addCookie(valueCookie);
    response.addCookie(operatorCookie);

    response.sendRedirect("Calculator-cookie-Redirect.html");
}
```

전달인자 없이 페이지 Link 함

Forward

CalcCookieRedirect.java

```
// +, - 일 때는 cookie에 담음
if (operator.equals("+") || operator.equals("-")) {
    // 쿠키 생성
    Cookie valueCookie = new Cookie("val", String.valueOf(strValue));
    Cookie operatorCookie = new Cookie("oper", String.valueOf(operator));
    // 생성된 쿠키를 클라이언트로 보냄.
    response.addCookie(valueCookie);
    response.addCookie(operatorCookie);

    RequestDispatcher dispatcher
        = request.getRequestDispatcher("Calculator-cookie-Redirect.html");
    dispatcher.forward(request, response);
}
```

현재 페이지의 request, response 값을 전달 가능


3.2. 계산기 예제

계산기 예제

View 페이지

ExamCalcView.java

```
input {
    width : 50px;
    height : 50px;
}
.output {
    width : 200px;
    height : 50px;
    font-size : 24px;
    font-weight : bold;
    text-align : right;
    padding : 0px 5px;
}
</style>
</head>
<body>
<form action = "calc-dynamic" method = "post">
<table border = 1>
<tr>
<td colspan="4" class="output"><div></div></td>
</tr>
<tr>
<td colspan="4">
<input type="submit" name="operator" value="CE"/>
<input type="submit" name="operator" value="C"/>
<input type="submit" name="operator" value="BS"/>
<input type="submit" name="operator" value="/" />
</td>
</tr>
<tr>
<td colspan="4">
<input type="submit" name="value" value="7"/>
<input type="submit" name="value" value="8"/>
<input type="submit" name="value" value="9"/>
<input type="submit" name="operator" value="*" />
</td>
</tr>
<tr>
<td colspan="4">
<input type="submit" name="value" value="4"/>
<input type="submit" name="value" value="5"/>
<input type="submit" name="value" value="6"/>
<input type="submit" name="operator" value="-" />
</td>
</tr>
<tr>
<td colspan="4">
<input type="submit" name="value" value="1"/>
<input type="submit" name="value" value="2"/>
<input type="submit" name="value" value="3"/>
<input type="submit" name="operator" value="+" />
</td>
</tr>
<tr>
<td colspan="2"><input type="submit" name="value" value="0"/>
<td colspan="2"><input type="submit" name="operator" value="=" />
</td>
</tr>
</table>
</form>
</body>
</html>
```



Application 소스

ExamCalcLogic.java

```
@WebServlet("/calc-dynamic-01")
public class Exam02Dynamic01 extends HttpServlet {
    protected void service (HttpServletRequest request, HttpServletResponse response ) {
        Cookie[] cookies = request.getCookies();
        String value = request.getParameter("value");
        String operator = request.getParameter("operator");
        String dot = request.getParameter("dot");
        String exp = "";
        if (cookies != null)
            for (Cookie c : cookies)
                if (c.getName().equals("exp"))
                    exp = c.getValue(); break;
        if (operator != null && operator.equals("=")) {
            ScriptEngine engine = new ScriptEngineManager().getEngineByName("nashorn");
            System.out.println("engine : " + engine);
            try {
                exp = String.valueOf(engine.eval(exp));
            } catch (ScriptException e) {
                e.printStackTrace();
            }
        }
        else if (operator != null && operator.equals("C")) {
            exp = "";
        }
        // ...
    }
}
```

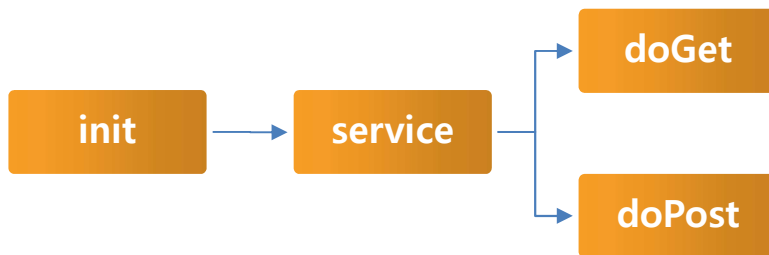
최종 완성 필요함.

3.3. service 메서드 활용

- 부모의 service 메서드를 override 하여 doGet / doPost 활용 가능함.
- 위/아래 모두 같은 내용임.



Service 메서드 override



```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
  <form action="CalculatorDo" method="get">
    <input type="submit" value="GET 요청" />
  </form>
  <form action="CalculatorDo" method="post">
    <input type="submit" value="POST 요청" />
  </form>
</body>
</html>
```

Calculator-do.html



```
@WebServlet("/CalculatorDo")
public class CalculationDo extends HttpServlet{
    public void service (HttpServletRequest request, HttpServletResponse response){
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        /*
        if (request.getMethod().equals("GET")) {
            System.out.println("GET 요청이 왔습니다.");
        }
        else if (request.getMethod().equals("POST")) {
            System.out.println("POST 요청이 왔습니다.");
        }
        */

        super.service(request, response);
    }
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        System.out.println("GET 요청이 왔습니다.");
    }
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        System.out.println("POST 요청이 왔습니다.");
    }
}
```

CalculatorDo.java