

2장. 서블릿(Servlet)



HttpServlet



서블릿(Servlet)

■ 서블릿(Servlet)

- 자바(Java) 기반의 웹 프로그래밍 기술로, 웹 서버에서 실행된다.
- 클라이언트(보통 웹 브라우저)의 요청(Request)을 처리하고 응답(Response)을 만들어 반환하는 서버 사이드 프로그램이다.
- **Servlet = Server + Applet**
- 웹 서버에서 실행되는 자바 클래스로, 클라이언트의 요청을 받아 동적 웹 페이지(DB 조회, 계산, 사용자 입력 처리 등)를 생성한다.
- 일반적으로 HTTP 프로토콜 기반으로 동작하며, 가장 대표적인 구현체는 **HttpServlet**이다.



서블릿(Servlet)

■ 서블릿의 장점

- 자바 언어 기반 → 객체지향적이고 안정적
- 플랫폼 독립적 → JVM만 있으면 어디서든 실행 가능
- 강력한 라이브러리 & API 지원 (DB, 네트워크 등)
- JSP, Spring Framework 같은 웹 기술의 기반

■ 서블릿의 한계

- 순수 서블릿으로 HTML을 출력하려면 `out.println("<h1>...</h1>")` 같은 코드가 복잡해짐
- 뷰(View) 처리에 불편 → JSP와 함께 사용하면서 MVC 패턴이 도입됨
- 이후 Spring, Spring Boot 같은 프레임워크들이 발전하면서 더 효율적인 개발 방식이 확산됨



■ Java EE API-> 서블릿(Servlet)

javax.servlet

Interface Servlet

All Known Subinterfaces:

HttpJspPage, JspPage

All Known Implementing Classes:

FacesServlet, GenericServlet, HttpServlet

javax.servlet.http

Class HttpServlet

java.lang.Object

javax.servlet.GenericServlet

javax.servlet.http.HttpServlet

All Implemented Interfaces:

Serializable, Servlet, ServletConfig



서블릿(Servlet)

▪ HttpServlet 이란?

- HttpServlet은 Servlet의 대표적인 구현 클래스이다.
- 클라이언트와 서버 간에 HTTP 프로토콜 기반의 요청과 응답을 처리하는 데 최적화된 서블릿입니다.
- javax.servlet.http.HttpServlet (Servlet 3.x까지) 또는 **jakarta.servlet.http.HttpServlet (Servlet 4.x 이상, Jakarta EE)** 패키지에 속해 있습니다.
- 보통 직접 Servlet 인터페이스를 구현하지 않고, HttpServlet을 **상속** 받아 필요한 메서드만 오버라이딩해서 사용합니다.



서블릿(Servlet)

▪ HttpServlet의 주요 메서드

- doGet() → GET 요청 처리 (브라우저 주소창 요청, 링크 클릭 등)
- doPost() → POST 요청 처리 (폼 제출, JSON 데이터 전송 등)

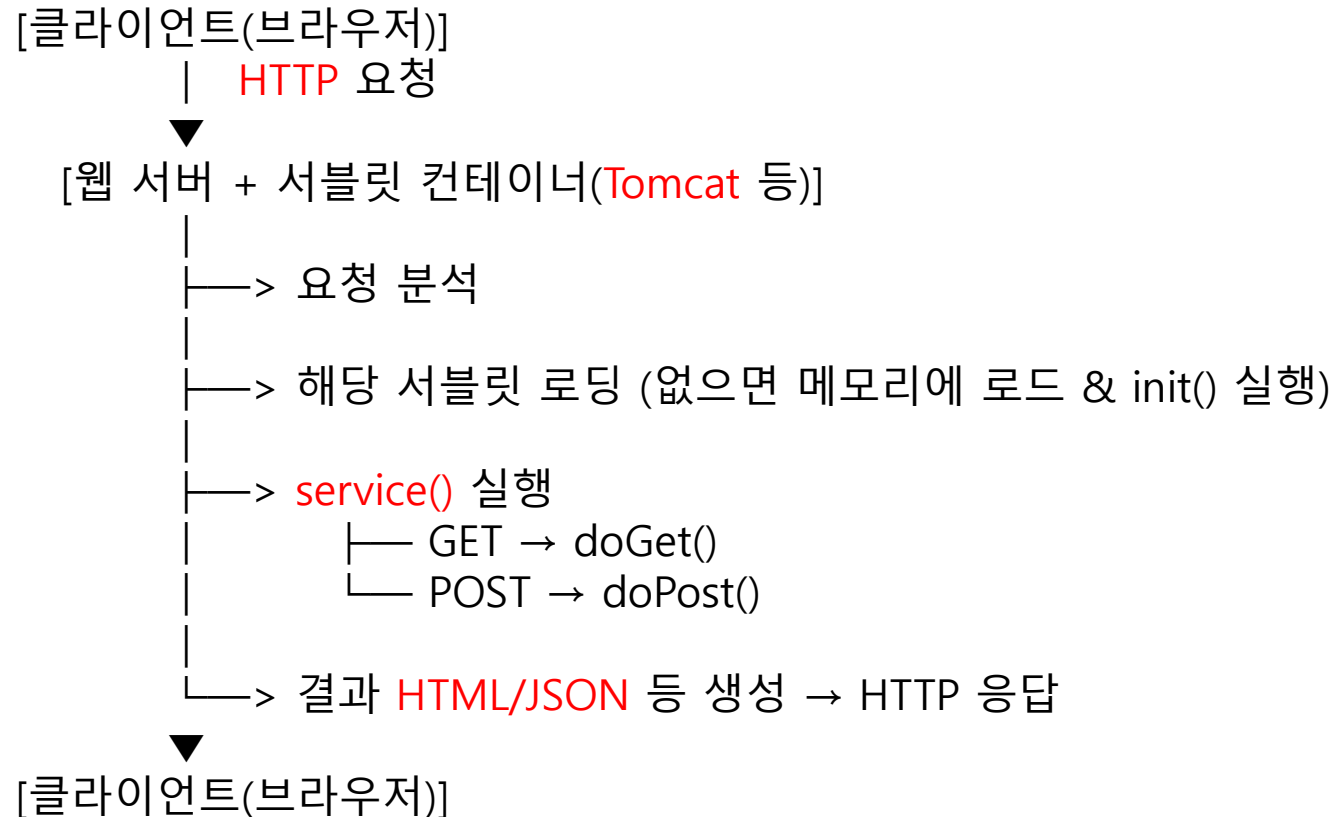
```
protected void doGet(HttpServletRequest req, HttpServletResponse resp)  
    throws ServletException, IOException;
```

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp)  
    throws ServletException, IOException;
```



서블릿(Servlet)

■ Servlet의 실행 흐름



서블릿(Servlet)

- Servlet 클래스 만들기

사용자 정의 서블릿 클래스 만들기



서블릿 생명주기 메서드(`init()`, `destroy()`)



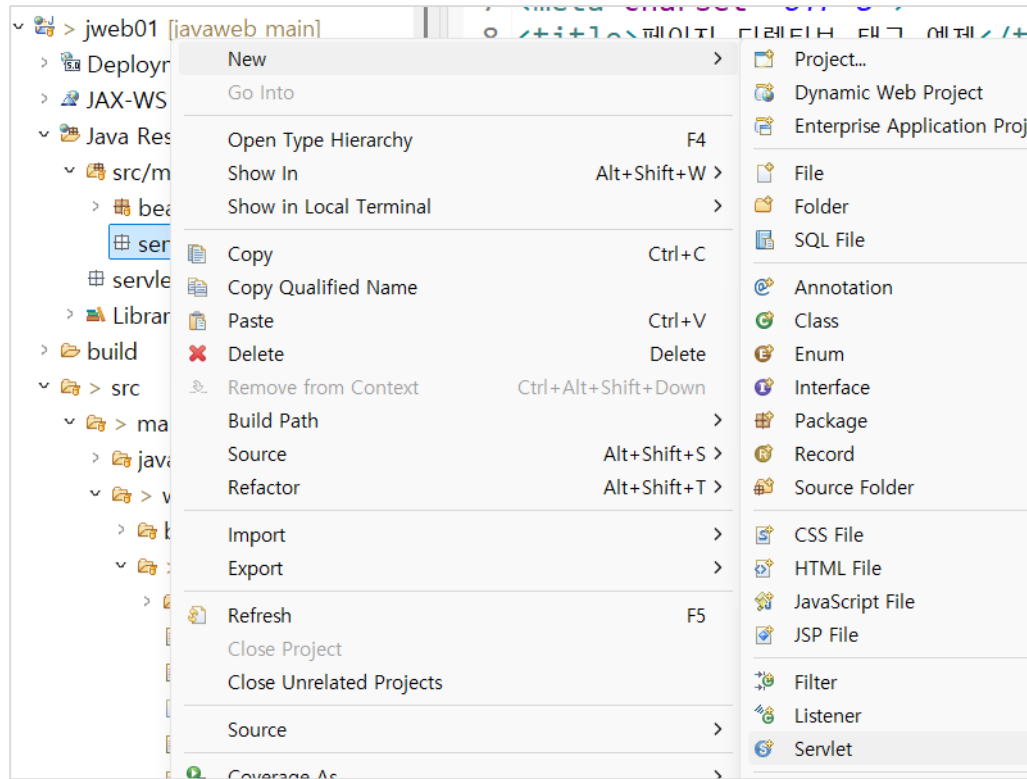
`service()`, `doGet()`, `doPost()`



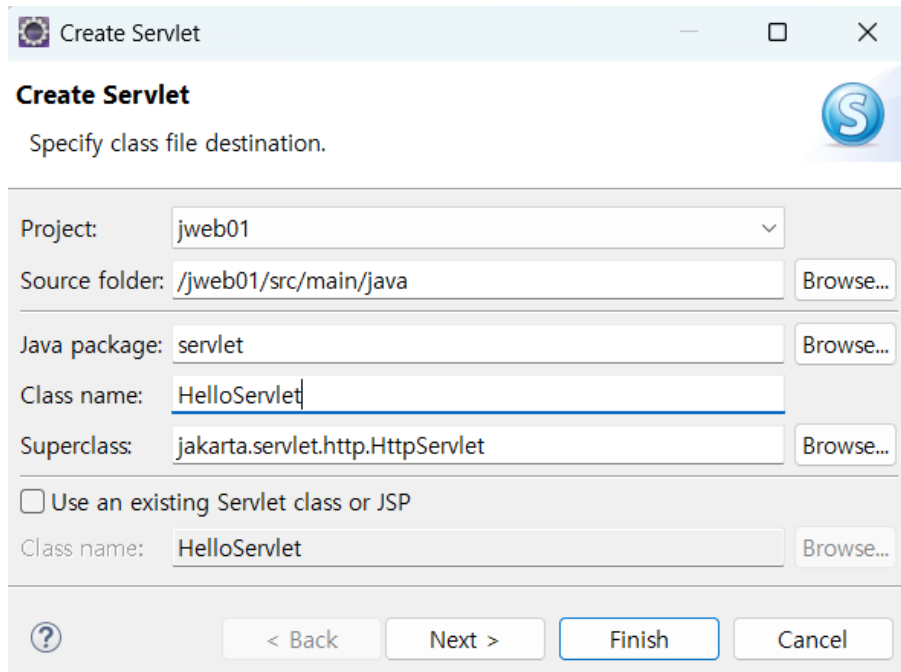
웹 브라우저에서 서블릿 매핑이름으로 요청

- 애너테이션(@)을 이용한 url 매핑

Java package 우클릭(메뉴) > New > Servlet



■ HelloWorld 생성



Create Servlet

Specify class file destination.

Project: jweb01

Source folder: /jweb01/src/main/java Browse...

Java package: servlet Browse...

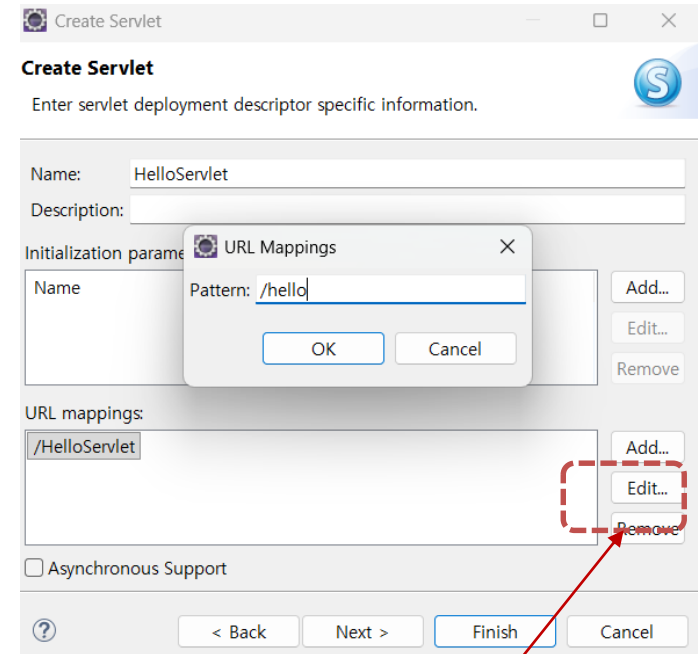
Class name: HelloWorld

Superclass: jakarta.servlet.http.HttpServlet Browse...

☐ Use an existing Servlet class or JSP

Class name: HelloWorld Browse...

? < Back Next > Finish Cancel



Create Servlet

Enter servlet deployment descriptor specific information.

Name: HelloWorld

Description:

Initialization parameters

Name

Pattern: /hello OK Cancel

URL mappings:

| Name | Pattern | Operations |
|---------------|---------|--|
| /HelloServlet | | Add... Edit... Remove |

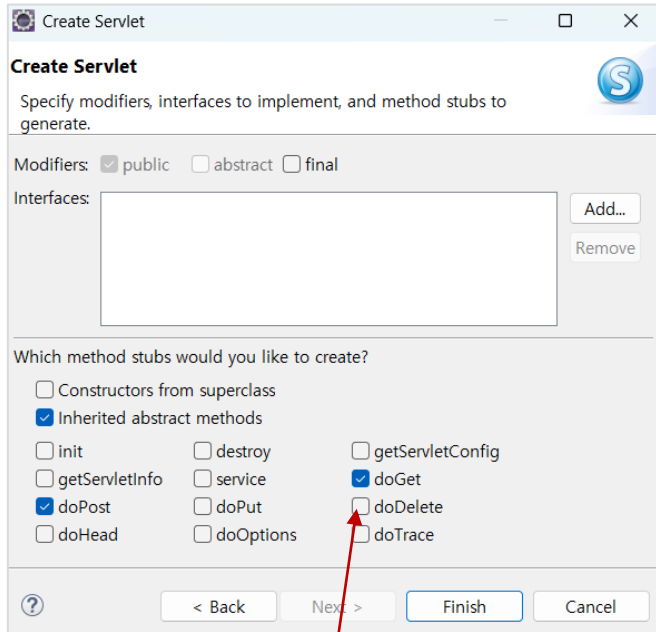
☐ Asynchronous Support

? < Back Next > Finish Cancel

edit > "/hello"로 변경

서블릿 만들기

■ HelloServlet 생성



The 'Create Servlet' dialog box is shown. It has a title bar with a question mark, a close button, and a maximize button. The main area is titled 'Create Servlet' and contains the instruction 'Specify modifiers, interfaces to implement, and method stubs to generate.' Below this, there are sections for 'Modifiers' (with checkboxes for public, abstract, and final), 'Interfaces' (with an 'Add...' button and a 'Remove' button), and 'Which method stubs would you like to create?'. The 'Inherited abstract methods' section is checked. Under this section, there are three columns of checkboxes: 'init', 'destroy', 'getServletConfig', 'doPost', 'doPut', 'doDelete', 'doHead', 'doOptions', and 'doTrace'. The 'doGet' checkbox is checked. At the bottom, there are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

필요 항목 체크함

웹 브라우저에 요청하는 매핑 이름

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.getWriter().append("Served at: ").append(request.

    protected void doPost(HttpServletRequest request, HttpServletResponse response) {
        doGet(request, response);
    }
}
```



서블릿의 응답 처리

- 서블릿 응답 실습



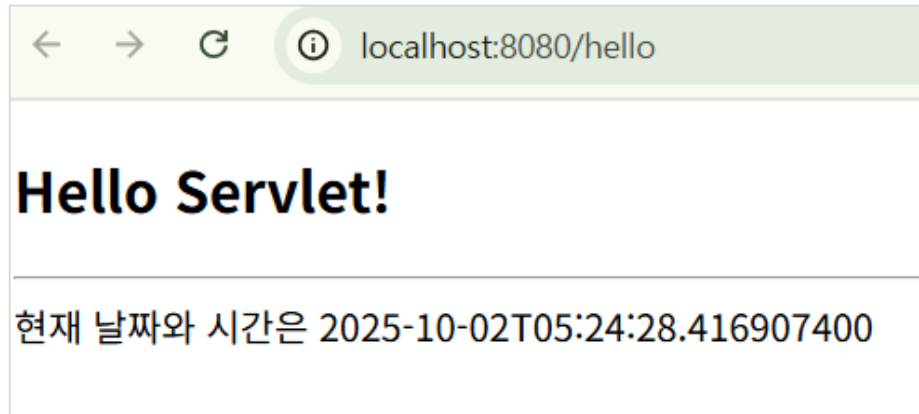
MIME-TYPE

톰캣 컨테이너에 미리 설정해 놓은 데이터 종류이다.

서버에서 웹 브라우저로 데이터를 전송할때 데이터 종류를 지정해서 전송한다.



- 서블릿 응답 실습



■ 서블릿 응답 실습

```
@WebServlet("/hello") // URL 패턴 매핑 (http://localhost:8080/프로젝트명/hello)
public class HelloServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        //응답 콘텐츠 타입 설정(type, 한글인코딩)
        response.setContentType("text/html; charset=UTF-8");

        //출력 스트림
        PrintWriter out = response.getWriter();

        //HTTP 응답
        out.println("<!DOCTYPE html><html>");
        out.println("<head><title>Hello Servlet</title></head>");
        out.println("<body><h2>Hello Servlet!</h2><hr>");
        out.println("현재 날짜와 시간은 " + LocalDateTime.now());
        out.println("</body></html>");
    }
```

HttpServletRequest로 요청 실습

■ HttpServletRequest란?

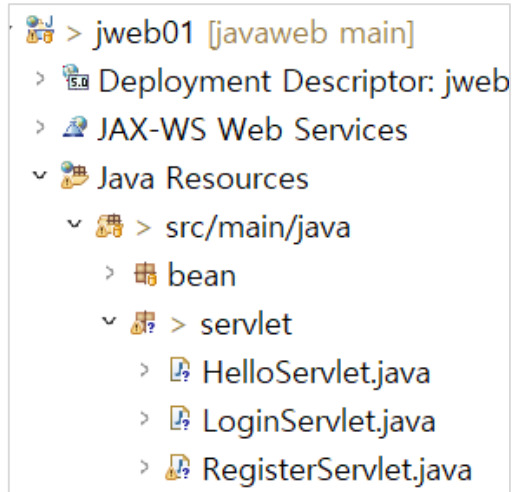
- 웹 서버에 들어오는 HTTP 요청에 대한 모든 정보를 담고 있는 객체이다.
- 클라이언트의 요청을 받아 서버가 처리할 수 있도록 HTTP 요청 메시지를 파싱하여 필요한 데이터를 제공한다.
- 클라이언트의 요청 메시지, 요청 헤더, 요청 파라미터 등을 조회하는 데 사용된다.

■ HttpServletRequest 주요 메서드

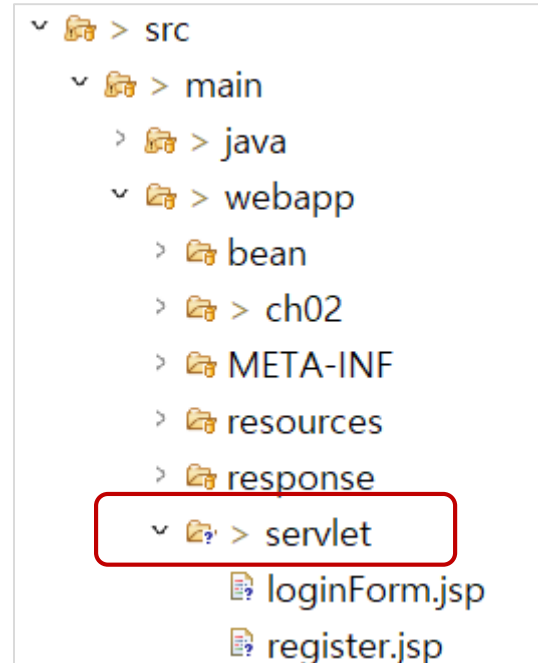
| 메소드 | 반환유형 | 설 명 |
|---------------------------------|-----------------------|--|
| getParameter(String name) | String | 요청 파라미터 이름이 name인 값을 전달받음 |
| getParameterValues(String name) | String[] | 모든 요청 파라미터 이름이 name인 값을 배열 형태로 전달 받음 |
| getParameterNames() | Java.util.Enumeration | 모든 요청 파라미터의 이름과 값을 Enumeration 객체 타입으로 전달 받음 |



■ 프로젝트 구조(java, webapp)



java



jsp

회원 가입 실습

■ 회원 가입

localhost:8080/servlet/register.jsp

회원 가입

이름 :

이메일 :



localhost:8080/register

회원 등록 결과

이름: 김기용

이메일: today@space.com

[다시 입력](#)

```
> jweb01 [javaweb main]
> Deployment Descriptor: jweb
> JAX-WS Web Services
v Java Resources
  v > src/main/java
    > bean
    v > servlet
      > HelloServlet.java
      > LoginServlet.java
      > RegisterServlet.java
```



■ 회원 가입 폼 – register.jsp

✓ 이 파일에서 톰캣 서버 실행함

요청 url(매핑)

```
<h2>회원 가입</h2>
<form action="/register" method="post">
  <ul>
    <li>
      <label for="uname">이름 : </label>
      <input type="text" id="uname" name="uname">
    </li>
    <li>
      <label for="email">이메일 : </label>
      <input type="text" id="email" name="email">
    </li>
    <li>
      <input type="submit" value="가입">
    </li>
  </ul>
</form>
```

servlet/register.jsp

■ RegisterServlet

```
@WebServlet("/register") //http://localhost:8080/register
public class RegisterServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    <method 방식이 post이므로 doPost() 사용함>
    protected void doPost(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {

        //request.setCharacterEncoding("utf-8"); //한글 깨짐(인코딩 설정)

        //요청 입력값 받기
        String name = request.getParameter("uname");
        String email = request.getParameter("email");

        //응답 설정
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();

        out.println("<!DOCTYPE html><html>");
        out.println("<head><title>회원 가입</title></head>");
        out.println("<body><h2>회원 등록 결과</h2><hr>");
        out.println("<p>이름: " + name + "</p>");
        out.println("<p>이메일: " + email + "</p>");
        out.println("<a href='/servlet/register.jsp'>다시 입력</a>");
        out.println("</body></html>");
    }
}
```

로그인 실습

■ 서블릿 응답 실습

아래와 같이 실행되도록 구현하세요.

- ✓ Login.jsp
- ✓ LoginServlet.java
- ✓ 서블릿 매핑: /login



- 여러 개의 name 요청 처리

회원 가입

이름 :

이메일 :

과목: ☒ java ☐ jsp ☒ springboot



회원 등록 결과

이름: 김기용

이메일: today@space.com

선택한 과목: java

선택한 과목: springboot

[다시 입력](#)

■ 여러 개의 name 요청 처리

```
<h2>회원 가입</h2>
<form action="/register2" method="post">
  <ul>
    <li>
      <label for="uname">이름 : </label>
      <input type="text" id="uname" name="uname">
    </li>
    <li>
      <label for="email">이메일 : </label>
      <input type="text" id="email" name="email">
    </li>
    <li>
      <label for="subject">과목: </label>
      <input type="checkbox" name="subject" value="java" checked>java
      <input type="checkbox" name="subject" value="jsp">jsp
      <input type="checkbox" name="subject" value="springboot">springboot
    </li>
    <li>
      <input type="submit" value="가입">
    </li>
  </ul>
</form>
```

■ 여러 개의 name 요청 처리

```
@WebServlet("/register2") //http://localhost:8080/register
public class RegisterServlet2 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        //request.setCharacterEncoding("utf-8"); //한글 깨짐(인코딩 설정)

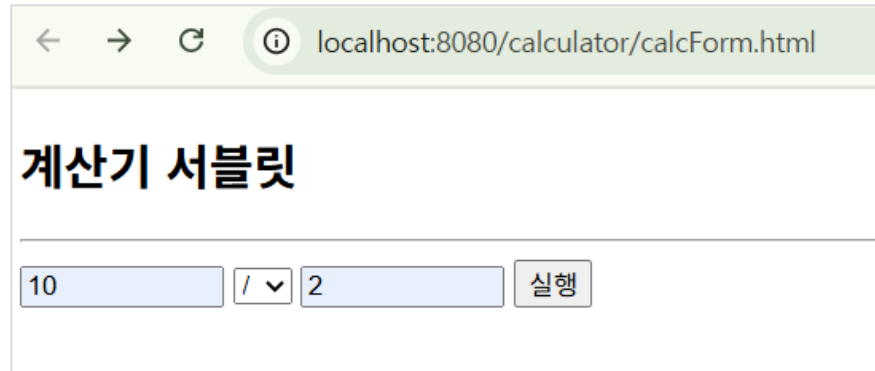
        //요청 입력값 받기
        String name = request.getParameter("uname");
        String email = request.getParameter("email");
        String[] subject = request.getParameterValues("subject");

        //응답 설정
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();

        out.println("<!DOCTYPE html><html>");
        out.println("<head><title>회원 가입</title></head>");
        out.println("<body><h2>회원 등록 결과</h2><hr>");
        out.println("<p>이름: " + name + "</p>");
        out.println("<p>이메일: " + email + "</p>");
        for(String subj : subject) {
            out.println("<p>선택한 과목: " + subj + "</p>");
        }
    }
```

간단한 계산기

- 정수형 계산기



A screenshot of a web browser window. The address bar shows 'localhost:8080/calculator/calcForm.html'. The page title is '계산기 서블릿'. Below the title is a form with two input fields: the first contains '10' and the second contains '2'. Between the fields is a dropdown menu showing a division symbol '/'. To the right of the second input field is a button labeled '실행' (Execute).



A box titled '정수형 계산기' (Integer Calculator). Below the title, it displays the result: '계산 결과: 5' (Calculation result: 5).

간단한 계산기

- 정수형 계산기

```
<h2>계산기 서버릿</h2>
<hr>
<form action="/calc" method="post">
  <input type="text" name="n1" size=10>
  <select name="op">
    <option>+</option>
    <option>-</option>
    <option>*</option>
    <option>/</option>
  </select>
  <input type="text" name="n2" size="10">
  <input type="submit" value="실행">
</form>
```

calculator/calcForm.jsp



■ 정수형 계산기

```
@WebServlet("/calc")
public class CalcServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) {
        int n1 = Integer.parseInt(request.getParameter("n1"));
        int n2 = Integer.parseInt(request.getParameter("n2"));
        String op = request.getParameter("op");

        int result = 0;

        switch(op) {
            case "+":
                result = n1 + n2; break;
            case "-":
                result = n1 - n2; break;
            case "*":
                result = n1 * n2; break;
            case "/":
                result = n1 / n2; break;
        }
    }
}
```

servlet.CalcServlet.java



간단한 계산기

■ 정수형 계산기

```
response.setContentType("text/html; charset=utf-8");

PrintWriter out = response.getWriter();
out.append("<html><body><h2>계산기 서버릿</h2><hr>")
    .append("계산 결과: " + result + "</body></html>");
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
    doGet(request, response);
}
```