



## 02. 서블릿

👤 생성자	👤 박지민
🕒 생성 일시	@2025년 3월 19일 오전 9:21

### ✅ 프로젝트 생성

#### 🌐 Spring Initializr 접속

- 사이트: <https://start.spring.io>

#### ⚙️ 설정

- **Project:** Gradle - Groovy
- **Language:** Java
- **Spring Boot:** 3.x.x
- **Group:** `hello`
- **Artifact:** `servlet`
- **Name:** `servlet`
- **Package name:** `hello.servlet`
- **Packaging:** `WAR` (중요! JSP 실행을 위해 필요)
- **Java 버전:** 17 또는 21
- **Dependencies:**
  - Spring Web
  - Lombok

### ⚠️ Spring Boot 3.x 주의사항

1. **Java 17 이상** 사용 필수
2. \* `javax` → `jakarta` \*로 패키지 변경됨

### 3. H2 데이터베이스 사용 시 2.1.214 이상 버전 필요

## ✓ 동작 확인

- `ServletApplication.main()` 실행
- 브라우저에서 <http://localhost:8080> 접속
- **Whitelabel Error Page**가 나오면 정상 실행

## ✓ Lombok 설정

### 1. IntelliJ 플러그인 설치

- Preferences → Plugins → Lombok 검색 후 설치, 재시작

### 2. Annotation Processing 활성화

- Preferences → Build, Execution, Deployment → Compiler → Annotation Processors → `Enable annotation processing` 체크 후 재시작

### 3. 테스트 클래스에서 `@Getter`, `@Setter` 등 확인

## ✓ 서블릿 등록하기

### 🔧 서블릿 자동 등록 설정

`ServletApplication.java`

```
@ServletComponentScan // 서블릿 자동 등록
@SpringBootApplication
public class ServletApplication {
    public static void main(String[] args) {
        SpringApplication.run(ServletApplication.class, args);
    }
}
```

### 💡 HelloServlet 생성

`hello.servlet.basic>HelloServlet.java`

```
@WebServlet(name = "helloServlet", urlPatterns = "/hello")
public class HelloServlet extends HttpServlet {
```

```

@Override
protected void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    System.out.println("HelloServlet.service");
    System.out.println("request = " + request);
    System.out.println("response = " + response);

    String username = request.getParameter("username");
    System.out.println("username = " + username);

    response.setContentType("text/plain");
    response.setCharacterEncoding("utf-8");
    response.getWriter().write("hello " + username);
}
}

```

## ✓ 테스트

- URL: `http://localhost:8080/hello?username=world`
- 결과: `hello world`
- 콘솔 로그:

```

HelloServlet.service
request = org.apache.catalina.connector.RequestFacade@...
response = org.apache.catalina.connector.ResponseFacade@...
username = world

```



## HTTP 요청 로그 확인

`application.properties` 설정 추가:

```
logging.level.org.apache.coyote.http11=trace
```

→ 서버 재시작 후 HTTP 요청 로그 확인 가능

## 참고

- **내장 톰캣 사용:** 별도로 설치할 필요 없이 내장 서버로 서블릿 실행 가능
- **Content-Length**는 톰캣이 자동 생성
- **주의:** `trace` 로그는 개발 환경에서만 사용! 운영 환경에서는 성능 저하 우려 있음

## 1. Welcome 페이지 구성

### ✓ 목적

개발 중 참고할 수 있는 **학습 링크 페이지**를 구성

### ✓ 위치

```
src/main/webapp/index.html
```

### ✓ 예시 코드

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Index</title>
</head>
<body>
<h1>Servlet 학습 - Welcome Page</h1>
<ul>
  <li><a href="/basic.html">서블릿 basic</a></li>
</ul>
</body>
</html>
```

## 2. 학습 메뉴 페이지 (basic.html)

```
src/main/webapp/basic.html
```

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>서블릿 기본</title>
</head>
<body>
<h1>서블릿 학습 메뉴</h1>
<ul>
  <li>hello 서블릿
    <ul>
      <li><a href="/hello?username=servlet">hello 서블릿 호출</a></li>
    </ul>
  </li>
  <li>HttpServletRequest
    <ul>
      <li><a href="/request-header">기본 사용법, Header 조회</a></li>
      <li><a href="/request-param?username=hello&age=20">GET - 쿼리
파라미터</a></li>
    </ul>
  </li>
  <li>HttpServletResponse
    <ul>
      <li><a href="/response-header">기본 사용법, Header 조회</a></li>
      <li><a href="/response-html">HTML 응답</a></li>
      <li><a href="/response-json">HTTP API JSON 응답</a></li>
    </ul>
  </li>
</ul>
</body>
</html>

```

## 3. HttpServletRequest 개요

### 역할

HTTP 요청 메시지를 개발자가 직접 파싱하지 않고, 스펙에 맞게 파싱한 결과를

`HttpServletRequest` 객체를 통해 제공

## 구조 (예시 메시지)

```
POST /save HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded

username=kim&age=20
```

## 4. `RequestHeaderServlet` 실습

```
@WebServlet(name = "requestHeaderServlet", urlPatterns = "/request-header")
public class RequestHeaderServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        printStartLine(request);
        printHeaders(request);
        printHeaderUtils(request);
        printEtc(request);
    }
}
```

### 4-1. START LINE 정보

```
request.getMethod();    // GET, POST
request.getProtocol();   // HTTP/1.1
request.getScheme();     // http, https
request.getRequestURL(); // 전체 URL
request.getRequestURI(); // URI
request.getQueryString(); // username=hello
request.isSecure();      // HTTPS 여부
```

## 4-2. Header 전체 조회

```
request.getHeaderNames().asIterator()
    .forEachRemaining(name → System.out.println(name + ": " + request.g
etHeader(name)));
```

## 4-3. Header 편의 메서드

```
request.getServerName(); // Host
request.getServerPort(); // Port

request.getLocales(); // 언어 목록
request.getLocale(); // 가장 우선 언어

request.getCookies(); // 쿠키 배열

request.getContentType(); // Content-Type
```

## 4-4. 기타 정보

```
request.getRemoteAddr(); // 클라이언트 IP
request.getRemotePort(); // 클라이언트 포트
request.getLocalAddr(); // 서버 IP
request.getLocalPort(); // 서버 포트
```

 **참고:** IPv4 사용 설정

```
-Djava.net.preferIPv4Stack=true
```

## 5. 요청 데이터 전송 방식 요약

방식	설명	예시
GET (쿼리 파라미터)	URL에 데이터 포함	<code>/search?keyword=java</code>
POST - HTML Form	Form 파라미터 형식으로 Body에 전달	<code>application/x-www-form-urlencoded</code>

POST/PUT - API

JSON, XML 등 Body에 직접 담아 전송

application/json

## 6. 쿼리 파라미터 조회 실습

### 테스트 URL

http://localhost:8080/request-param?username=hello&age=20

http://localhost:8080/request-param?username=hello&username=kim&age=20

### 주요 메서드

```
request.getParameter("username");    // 단일 파라미터  
request.getParameterValues("username"); // 복수 파라미터  
request.getParameterNames();        // 모든 이름 조회  
request.getParameterMap();          // 이름 -> 배열 Map
```



### RequestParamServlet 코드

```
@WebServlet(name="RequestParamServlet", urlPatterns = "/request-param")  
public class RequestParamServlet extends HttpServlet {  
    @Override  
    protected void service(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        System.out.println("[전체 파라미터 조회]");  
        request.getParameterNames().asIterator()  
            .forEachRemaining(name -> System.out.println(name + "=" + request.getParameter(name)));  
  
        System.out.println("[단일 파라미터 조회]");  
        String username = request.getParameter("username");  
        String age = request.getParameter("age");
```



```

System.out.println("username = " + username);
System.out.println("age = " + age);

System.out.println("[이름이 같은 복수 파라미터 조회]");
String[] usernames = request.getParameterValues("username");
for (String name : usernames) {
    System.out.println("username = " + name);
}

response.getWriter().write("ok");
}
}

```

## ⚠ 주의

`username=hello&username=kim`

→ `request.getParameter("username")` 은 첫 번째 값만 반환

→ 복수 값 조회 시 `getParameterValues()` 사용 필수

## 🧰 7. 부가기능 요약

### 📁 임시 저장소

- 요청 범위 내 데이터 저장

```

request.setAttribute("key", value); // 저장
request.getAttribute("key");        // 조회

```

### 🔑 세션 기능

- 로그인 등 사용자 상태 유지

```
HttpSession session = request.getSession(); // 기본: true
```

### 📖 요청 로그 보기

```
# application.properties
logging.level.org.apache.coyote.http11=trace
```



## 1. POST HTML Form

### 특징

- 주 사용처: 회원가입, 로그인, 상품 주문 등
- **Content-Type:** `application/x-www-form-urlencoded`
- **Message Body 예시:** `username=hello&age=20`



### HTML Form 예시

파일 위치: `src/main/webapp/basic/hello-form.html`

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Form Test</title>
</head>
<body>
<form action="/request-param" method="post">
  username: <input type="text" name="username" /><br/>
  age: <input type="text" name="age" /><br/>
  <button type="submit">전송</button>
</form>
</body>
</html>
```



### 주의사항

- 브라우저 캐시로 인해 HTML 변경 사항이 바로 반영되지 않을 수 있음 → 새로고침 필요
- 서버를 재시작하지 않은 경우도 반영이 안 될 수 있음


## 브라우저 전송 내용 (개발자 도구 Network 탭 확인)

항목	값
요청 URL	http://localhost:8080/request-param
Method	POST
Content-Type	application/x-www-form-urlencoded
Message Body	username=hello&age=20

## 처리 방식

- GET 방식과 마찬가지로 쿼리 파라미터 조회 메서드 사용 가능

```
request.getParameter("username");  
request.getParameter("age");
```

 서버 입장에서 GET과 POST(Form)의 데이터 형식은 동일하므로, `request.getParameter()`를 사용하면 **형식 구분 없이** 사용할 수 있음.

## 2. Postman 테스트 팁

설정	설명
Method	POST
URL	http://localhost:8080/request-param
Body	x-www-form-urlencoded 선택 후 key-value 입력
Headers	Content-Type 자동 지정됨

## 3. HTTP 메시지 바디 - 텍스트 직접 읽기

### 목적

API 방식으로 메시지 바디에 텍스트 데이터를 직접 넣고 읽는 법을 학습

### 서블릿 예제

```
@WebServlet(name = "RequestBodyStringServlet", urlPatterns = "/request  
-body-string")  
public class RequestBodyStringServlet extends HttpServlet {  
    @Override
```

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

    ServletInputStream inputStream = request.getInputStream();
    String messageBody = StreamUtils.copyToString(inputStream, Standar
dCharsets.UTF_8);

    System.out.println("messageBody = " + messageBody);
    response.getWriter().write("ok");
}
}
```

### 요청 예시

- URL: `POST http://localhost:8080/request-body-string`
- Headers: `Content-Type: text/plain`
- Body:

hello

### 참고

- `request.getInputStream()` 은 byte 단위이므로 문자열로 변환할 땐 **Charset 지정 필수**
  - `StreamUtils.copyToString(..., StandardCharsets.UTF_8)`

## 4. HTTP 메시지 바디 - JSON

### 목적

실제 API 개발에서 가장 많이 사용하는 방식: JSON

### 요청 예시

- **Method:** POST
- **URL:** `http://localhost:8080/request-body-json`
- **Headers:** `Content-Type: application/json`

- **Body (raw → JSON 선택):**

```
{  
  "username": "hello",  
  "age": 20  
}
```

## **HelloData 클래스**

```
@Getter  
@Setter  
public class HelloData {  
    private String username;  
    private int age;  
}
```

| Lombok이 자동으로 getter/setter 생성

## **JSON 파싱 서블릿**

```
@WebServlet(name="requestBodyJsonServlet", urlPatterns="/request-body-json")  
public class RequestBodyJsonServlet extends HttpServlet {  
    private final ObjectMapper objectMapper = new ObjectMapper();  
  
    @Override  
    protected void service(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        ServletInputStream inputStream = request.getInputStream();  
        String messageBody = StreamUtils.copyToString(inputStream, StandardCharsets.UTF_8);  
  
        System.out.println("messageBody = " + messageBody);  
    }  
}
```

```

        HelloData helloData = objectMapper.readValue(messageBody, HelloData.class);

        System.out.println("helloData.username = " + helloData.getUsername());
        System.out.println("helloData.age = " + helloData.getAge());

        response.getWriter().write("ok");
    }
}

```

## 🧠 JSON → Java 객체 변환

- 사용 라이브러리: **Jackson** ( **ObjectMapper** )
- Spring Boot는 Jackson을 기본 의존성에 포함하고 있어 추가 설정 없이 사용 가능

## ✅ 출력 예시

```

messageBody = {"username": "hello", "age": 20}
helloData.username = hello
helloData.age = 20

```

## 🔄 정리 요약

구분	데이터 전송	Content-Type	처리 방식
GET	URL 쿼리 파라미터	없음	request.getParameter
POST - HTML Form	Form 데이터	application/x-www-form-urlencoded	request.getParameter
POST - API 텍스트	Body (text/plain)	text/plain	request.getInputStream + 변환
POST - API JSON	Body (JSON)	application/json	request.getInputStream + ObjectMapper

## ✅ 1. HttpServletResponse 개요

## 💡 주요 역할

서블릿에서 HTTP 응답 메시지를 생성하는 데 사용된다.

즉, 상태 코드, 헤더, 쿠키, 본문 등 응답 메시지를 구성하는 기능을 제공한다.

## 🔪 2. 기본 응답 구성 예제

### 📄 ResponseHeaderServlet

```
@WebServlet(name="ResponseHeaderServlet", urlPatterns = "/response-header")
public class ResponseHeaderServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        // [status-line]
        response.setStatus(HttpServletResponse.SC_OK); // 200

        // [response-headers]
        response.setHeader("Content-Type", "text/plain;charset=utf-8");
        response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate");
        response.setHeader("Pragma", "no-cache");
        response.setHeader("my-header", "hello");

        // [Header 편의 메서드]
        content(response);
        cookie(response);
        redirect(response);

        // [message body]
        PrintWriter writer = response.getWriter();
        writer.println("ok");
    }

    private void content(HttpServletResponse response) {
```

```

        response.setContentType("text/plain");
        response.setCharacterEncoding("utf-8");
        // response.setContentLength(2); // 생략 가능
    }

    private void cookie(HttpServletResponse response) {
        Cookie cookie = new Cookie("myCookie", "good");
        cookie.setMaxAge(600); // 600초
        response.addCookie(cookie);
    }

    private void redirect(HttpServletResponse response) throws IOException
    {
        response.sendRedirect("/basic/hello-form.html");
        // 자동으로 Status Code 302 + Location 헤더 설정됨
    }
}

```

### ✓ 3. 응답 메시지 구성 요소

항목	설명	메서드
Status Line	상태 코드 (200, 302 등)	<code>response.setStatus()</code>
Headers	응답 헤더 설정	<code>response.setHeader()</code> 등
Body	응답 본문	<code>response.getWriter().write()</code>

### ✓ 4. HTML 응답

#### ResponseHtmlServlet

```

@WebServlet(name = "responseHtmlServlet", urlPatterns = "/response-ht
ml")
public class ResponseHtmlServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {

```



```

response.setContentType("text/html");
response.setCharacterEncoding("utf-8");

PrintWriter writer = response.getWriter();
writer.println("<html>");
writer.println("<body>");
writer.println("<div>안녕?</div>");
writer.println("</body>");
writer.println("</html>");
}
}

```

### 주의

- Content-Type은 반드시 `text/html` 로 설정
- 브라우저에서 **페이지 소스 보기**로 HTML 구조 확인 가능

## 5. JSON 응답

### ResponseJsonServlet

```

@WebServlet(name = "responseJsonServlet", urlPatterns = "/response-json")
public class ResponseJsonServlet extends HttpServlet {
    private final ObjectMapper objectMapper = new ObjectMapper();

    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        // [1] Content-Type 설정
        response.setHeader("Content-Type", "application/json");
        response.setCharacterEncoding("utf-8");

        // [2] 응답할 데이터 객체 생성
        HelloData data = new HelloData();
    }
}

```

```

data.setUsername("kim");
data.setAge(20);

// [3] 객체 -> JSON 문자열 변환
String result = objectMapper.writeValueAsString(data);

// [4] 응답 메시지 출력
response.getWriter().write(result);
}
}

```

## HelloData 클래스

```

@Getter @Setter
public class HelloData {
    private String username;
    private int age;
}

```

## 결과 예시

```

{
  "username": "kim",
  "age": 20
}

```

## JSON 응답 주의사항

항목	설명
Content-Type	<code>application/json</code> (추가 파라미터 없이 설정해야 함)
문자 인코딩	JSON은 스펙상 UTF-8 기본, <code>charset=utf-8</code> 은 의미 없음
출력 방식	<code>response.getWriter()</code> 사용 시 charset이 추가될 수 있음 → 문제 발생 가능 시 <code>getOutputStream()</code> 사용

## JSON 관련 도구: Jackson

항목	설명
ObjectMapper	JSON ↔ 객체 변환
writeValueAsString(obj)	객체 → JSON 문자열
readValue(json, class)	JSON 문자열 → 객체

스프링 부트는 `ObjectMapper` 를 자동으로 제공하므로 별도 설정 없이 사용 가능

---