

JSP 2-1

[JSP 1-1](#)

● Servlet 개요 및 동작 원리

■ Servlet 이란?

서버에서 웹 페이지 등을 동적으로 생성하거나 데이터 처리를 수행하기 위해 자바로 작성된 프로그램.

- Java 코드 안에 HTML 태그 삽입
- Java 언어로 된 웹 어플리케이션 API
- HttpServlet을 상속하여 사용
- 클라이언트 요청을 처리하고 응답하는 자바 프로그램

 **Servlet은 클라이언트 요청을 처리하고 결과를 전송하는 규칙을 따른 자바 프로그램**

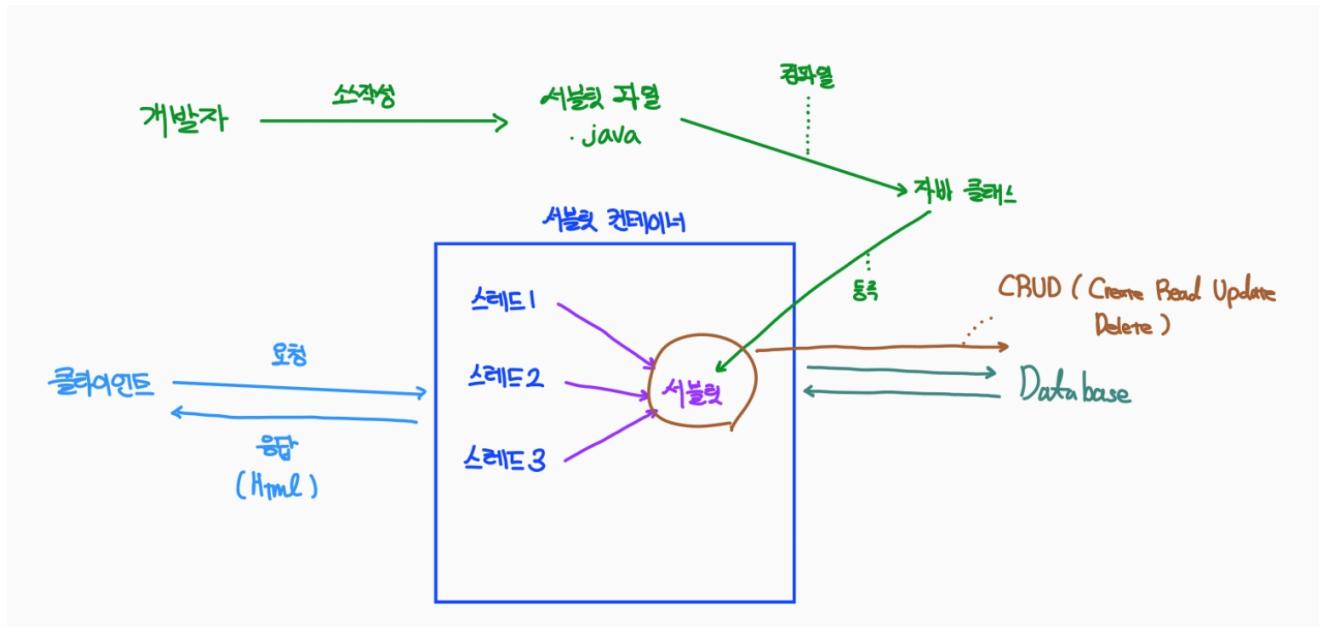
❖ Servlet 특징

- 동적 웹 어플리케이션 구현
- 파일 확장자: .java
- 클라이언트 요청에 따라 HTML 응답 생성
- Java Thread 기반으로 작동
- MVC 패턴에서 **Controller** 역할 수행

▣ Servlet 작동 순서

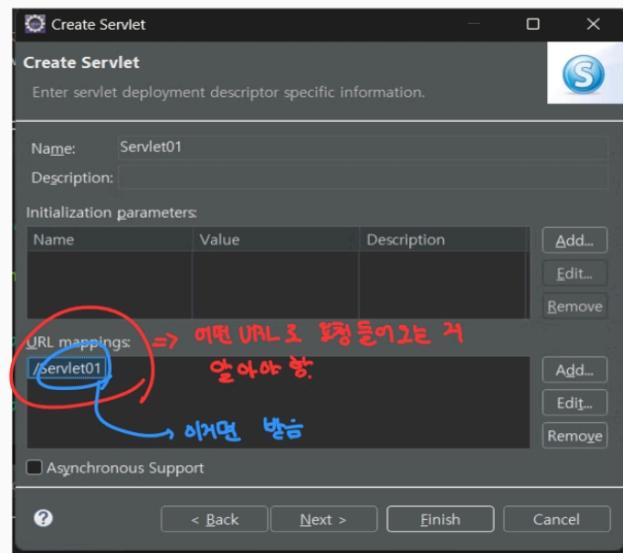
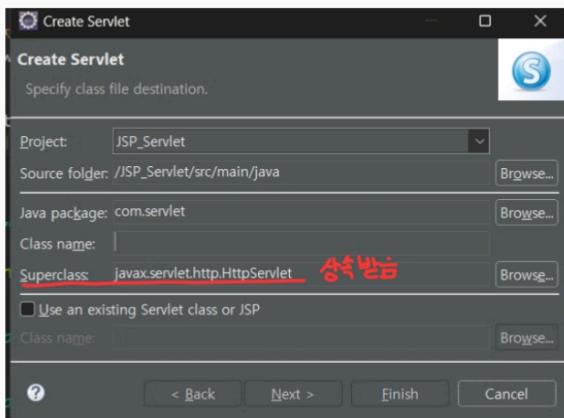
1. 클라이언트가 요청을 보냄
2. 서버는 Servlet 컨테이너 생성
3. 컨테이너는 HttpServletRequest, HttpServletResponse 객체 생성
4. 요청 URL에 해당하는 Servlet 탐색
5. 없으면 새 Servlet 객체 생성 후 스레드로 위임
6. 동적 HTML 페이지 생성하여 응답 객체에 저장
7. 응답 완료 후 두 객체는 소멸

 Java 기반 스레드로 작동하여 서버 부하가 적고 속도가 빠름 → **효율적인 서버 운영**



🛠️ Servlet 실습 이미지

Servlet 과정 생성 시



```
1 package com.servlet;
2
3 import java.io.IOException; // 어노테이션
4
5 @WebServlet("/Servlet01")
6 public class Servlet01 extends HttpServlet {
7     private static final long serialVersionUID = 1L;
8
9     public Servlet01() {
10        super();
11    }
12
13    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
14        // TODO Auto-generated method stub
15        response.getWriter().append("Served at: ").append(request.getContextPath());
16    }
17
18    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
19        // TODO Auto-generated method stub
20        doGet(request, response);
21    }
22
23 }
```

Servlet <--> GenericServlet <--> HttpServlet
« interface » { abstract }

Servlet 클래스는 HttpServlet 클래스 상속받음

☞ @WebServlet 어노테이션

어노테이션은 코드에 의미를 부여하는 메타데이터로, 코드 실행 흐름에 관여

- 주석처럼 사용되지만 실행 시 처리됨

- 코드 간결화, 재사용성 증가
- URL 매팅은 @WebServlet 또는 web.xml에 설정

```

Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk-11\bin
INFO: 웹 애플리케이션 디렉토리 [C:\JSP_class\JSP\apache-tomcat-9.0.58]
10월 28, 2024 10:19:38 오후 org.apache.catalina.startup.Bootstrap
INFO: 웹 애플리케이션 디렉토리 [C:\JSP_class\JSP\apache-tomcat-9.0.58]
10월 28, 2024 10:19:38 오후 org.apache.catalina.startup.Bootstrap
INFO: 웹 애플리케이션 디렉토리 [C:\JSP_class\JSP\apache-tomcat-9.0.58]
10월 28, 2024 10:19:38 오후 org.apache.catalina.startup.Bootstrap
INFO: 웹 애플리케이션 디렉토리 [C:\JSP_class\JSP\apache-tomcat-9.0.58]
10월 28, 2024 10:19:38 오후 org.apache.coyote.Abstract
INFO: 프로토콜 핸들러 ["http-nio-8090"]을(를) 시작합니다.
10월 28, 2024 10:19:38 오후 org.apache.catalina.startup.Bootstrap
INFO: 서버가 [1941] 밀리초 내에 시작되었습니다.
서블릿 클래스01입니다.

```

. 서버가 해석해 애플리케이션 구분하는 Path

① localhost:8090/JSP_Servlet/Servlet01
 포트번호 Context Path ↳ @WebServlet ("~/Servlet01")

@WebServlet("/hello")
 public class Servlet01

② localhost:8090/JSP_Servlet/hello ↲

다시 Servlet이 로하면 들어가지 않는다.

③ @WebServlet ("~/hello")로 바꾸었기 때문.

lic class Servlet01 extends HttpServlet {
 private static final long serialVersionUID = 1L;

: 접두어 키

개체로 byte로 변환하여 파일에 저장.

```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("서블릿 클래스01입니다.");
}
  
```

```

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

⌚ Request / Response 객체

- request : 클라이언트의 요청을 처리하는 객체 (예: 로그인 정보 등)
- response : 서버의 응답을 처리하는 객체 (예: 로그인 결과 전달)

```
• protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("서블릿 클래스01 입니다.");
    response.setCharacterEncoding("UTF-8");
    //UTF-8로 인코딩하여 웹 브라우저에게 전송한다.
    response.setContentType("text/html; charset=UTF-8");
    //웹 브라우저에게 명령한다고 보면된다
    //웹 브라우저에게 html 문서를 UTF-8로 해석하라는 뜻이다
    PrintWriter out = response.getWriter();
    //웹 브라우저에게 출력하기 위한 출력 스트림 얻기
    //out.print("");
    //out.println();
    //두 개의 차이점이 있다
    //웹 브라우저에서 줄 바꿈은 br태그라서 두 개의 차이점이 없다
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet start</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Hello Servlet</h1>");
    out.println("<h2>서블릿 시작합니다</h2>");
    out.println("</body>");
    out.println("</html>");
    out.close();
}
```

🔗 Servlet Mapping

Servlet 실행 시 연결되는 URL 패턴

방법 1: web.xml 사용

```
<welcome-file>default.htm</welcome-file>
</welcome-file-list>

<servlet>
    <!-- 임의의 이름을 만들어준다 -->
    <servlet-name>myServlet1</servlet-name>

    <!-- 등록할 서블릿 파일명을 패키지명을 포함하여 정확하게 입력한다 -->
    <servlet-class>com.servlet.Servlet01</servlet-class>

</servlet>

<servlet-mapping>
    <servlet-name>myServlet1</servlet-name>

    <!-- 주의: /로 시작해야한다 -->
    <url-pattern>/Hi</url-pattern>
</servlet-mapping>
</web-app>
```

방법 2: 어노테이션 사용

```
@WebServlet("/hello")
public class Servlet01
```

(web.xml 개요)

Web Application 동작 설정 파일

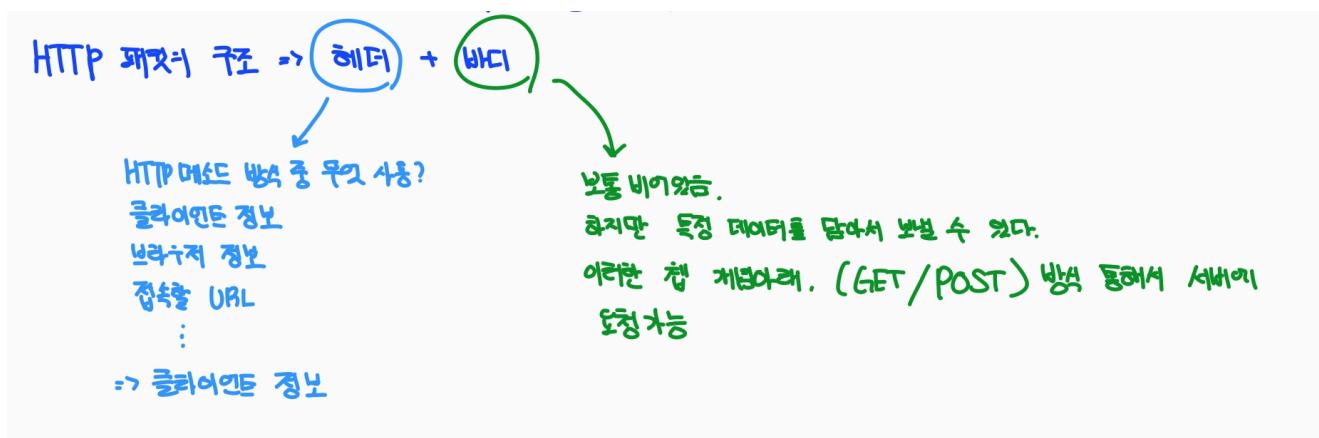
설정 내용

- 초기 parameter 설정
- Servlet/JSP 정의 및 맵핑
- MIME 타입
- 에러 페이지 처리
- 세션 유효시간

- Welcome 파일 지정
- 리스너, 필터 설정 등

HTTP 패킷

클라이언트가 서버로 보내는 요청 데이터



GET vs POST 방식

GET

- URL 뒤에 ? 와 & 로 파라미터 전송
- 헤더에 포함
- 데이터가 노출됨 → 보안에 취약

POST

- Body에 데이터 포함
- 대용량 데이터 전송 가능
- URL에 노출되지 않아 보안 상대적으로 안전

```

1 package com.servlet;
2
3 import java.io.IOException;
4
5
6 @WebServlet("/getPost")
7 public class Servlet02 extends HttpServlet
8 {
9     private static final long serialVersionUID = 1L;
10
11     protected void doGet(HttpServletRequest request, HttpServletResponse response)
12     {
13         System.out.println("doGet메소드입니다");
14     }
15
16     protected void doPost(HttpServletRequest request, HttpServletResponse response)
17     {
18         System.out.println("doPost메소드입니다");
19     }
20
21 }
22
23
24
25
26
27
28 }
29

```

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <h2>GET 방식</h2>
9
10<form action="/JSP_Servlet/getPost" method="get">
11     <input type="submit" value="GET 방식">
12 </form>
13
14 <hr>
15
16 <h2>POST 방식</h2>
17<form action="/JSP_Servlet/getPost" method="post">
18     <input type="submit" value="POST 방식">
19 </form>
20 </body>
21 </html>

```

실행

GET 방식 *내에 데이터가 전송되어 보안에 약하다*

POST 방식 *내에 데이터 담겨져 있지 않아 보안에 강하다*

Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk-11\bin\javaw.exe

```

10월 28, 2024 11:37:35 오후 org.apache.catalina.core.ContainerBase.[Catalina].INFO: 앱 애플리케이션 디렉토리 [C:\JSP_class\JSP]
10월 28, 2024 11:37:35 오후 org.apache.catalina.core.ContainerBase.[Catalina].INFO: 앱 애플리케이션 디렉토리 [C:\JSP_class\JSP]
10월 28, 2024 11:37:35 오후 org.apache.coyote.http11.Http11Protocol$Adapter.startInternal() 시작
10월 28, 2024 11:37:36 오후 org.apache.catalina.core.ContainerBase.[Catalina].INFO: 서버가 [2006] 밀리초 내에 시작되었습니다.
doGet메소드입니다
doPost메소드입니다

```

Request / Response 클래스

HttpServletRequest

- 요청 정보 전달 (헤더, 파라미터, 쿠키 등)

HttpServletResponse

- 응답 정보 구성 (Content-type, status 등)

Servlet Parameter

- form 제출 시 request로 전달된 데이터 추출

```
String id = request.getParameter("id"); // 단일 파라미터
String[] checkboxes = request.getParameterValues("chk"); // 복수 파라미터
Enumeration<String> names = request.getParameterNames(); // 전체 파라미터 이름
```

```
<body>
<h2>GET 방식</h2>
<form action="/JSP_Servlet/getPost" method="get">
    아이디: <input type="text" name="id"><br>
    비밀번호: <input type="text" name="pw"><br>
    <input type="submit" value="GET 방식"/>
</form>

<hr>

<h2>POST 방식</h2>
<form action="/JSP_Servlet/getPost" method="post">
    아이디: <input type="text" name="id"><br>
    비밀번호: <input type="text" name="pw"><br>
    <input type="submit" value="POST 방식"/>
</form>
```

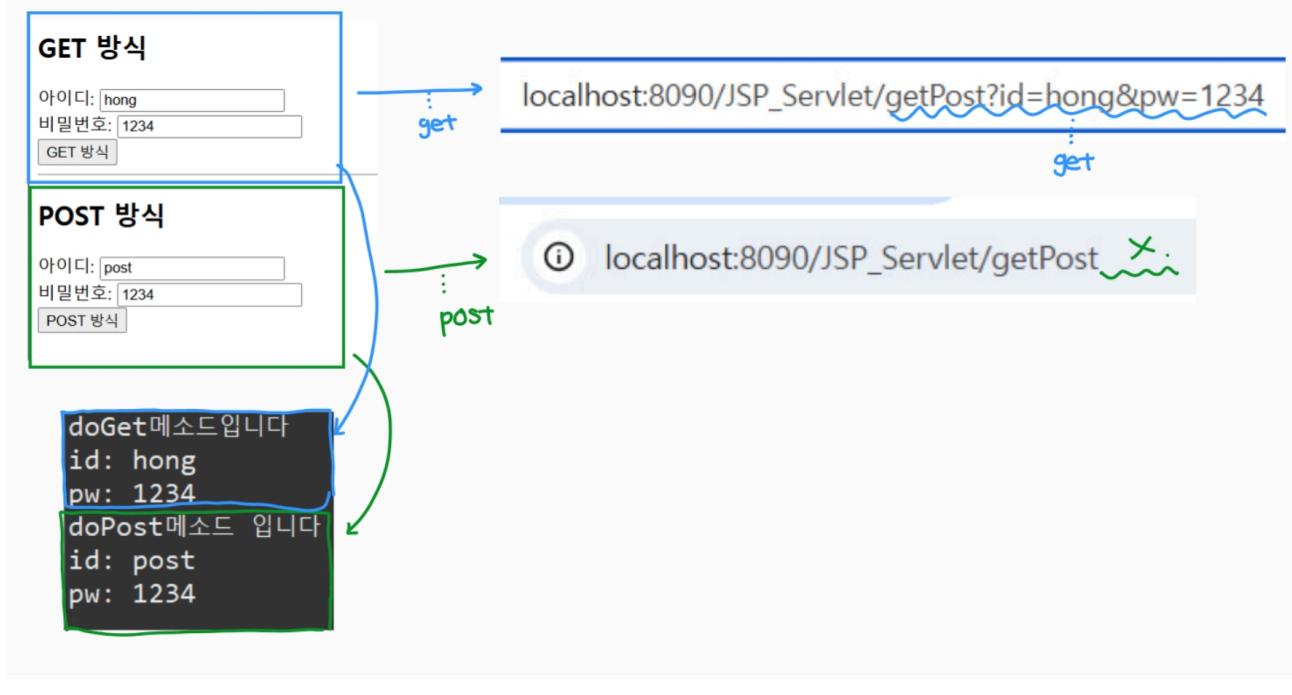
```
@WebServlet("/getPost")
public class Servlet02 extends HttpServlet
{
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        System.out.println("doGet메소드입니다");
        String id = request.getParameter("id"); // 넘어오는 데이터의 이름을 지정
        String pw = request.getParameter("pw");

        System.out.println("id: "+id);
        System.out.println("pw: "+pw);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        System.out.println("doPost메소드입니다");
        String id = request.getParameter("id"); // 넘어오는 데이터의 이름을 지정
        String pw = request.getParameter("pw");

        System.out.println("id: "+id);
        System.out.println("pw: "+pw);
    }
}
```



有 한글 파라미터 처리

- Tomcat 기본 문자 인코딩: ISO-8859-1 → 한글 깨짐 현상 발생

POST 방식 처리

```
request.setCharacterEncoding("UTF-8");
response.setCharacterEncoding("UTF-8");
```

```

request.setCharacterEncoding("UTF-8");
//post방식의 요청 한글처리

```

GET 방식 처리 (server.xml 수정)

```

Define a non-SSL/TLS HTTP/1.1 Connector
-->
<Connector URIEncoding="UTF-8" connectionTim
<!-- A "Connector" using the shared thread p
<!--

```

GET 방식의 요청 한글처리

Form 태그로 데이터 전송

Form 태그로 데이터 전송

```

memberform.jsp
<%
String id = request.getParameter("id");
String pw = request.getParameter("pw");
String name = request.getParameter("name");
String age = request.getParameter("age");
String hobby[] = request.getParameterValues("hobby");
String area = request.getParameter("area");
%>

```

```

memberform.java
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String name = request.getParameter("name");
    String age = request.getParameter("age");
    String hobby[] = request.getParameterValues("hobby");
    String area = request.getParameter("area");
}

```

```

id: Lee
pw: 1234
name: 이자바
age: 24
hobby: [등산, 게임]
area: seoul

```

