



# 서블릿(Servlet) 이란

서블릿(Servlet)은 서버에서 웹페이지 등을 동적으로 생성하거나 데이터 처리를 수행하기 위해 자바로 작성된 프로그램이다. Servlet은 java코드 안에 HTML태그가 삽입되며 자바언어로 되어있다. (.java가 확장자이다.)

자바언어를 웹어플리케이션에 조금 더 쉽게 개발하기 위해 만든 API이며 이 규약에 맞는 라이브러리나 클래스들을 상속 및 구현하여 만든 클래스이다.

서블릿(servlet)은 클라이언트 요청을 처리하고 그 결과를 다시 클라이언트에게 전송하는 구현 규칙을 지킨 자바프로그램(클래스) 이다.



# 서블릿(Servlet) 특징

서블릿(Servlet) 특징
동적 웹 어플리케이션 컴포넌트이다.
확장자 → .java 이다.
클라이언트 요청에 동적작동, 응답은 HTML 이용.
자바 Thread 를 이용하여 동작한다.
MVC 패턴에서 Controller로 이용된다.



# Servlet 작동순서

클라이언트에서 요청이 들어오면 서버에서는 서블릿 컨테이너를 만들고, 서블릿 컨테이너는 **HttpServletRequest, HttpServletResponse** 객체를 생성 한다.

서블릿 컨테이너는 request(요청) 를 분석하여 어떤 서블릿에 대한 요청인지를 찾아 해당하는 서블릿 클래스가 메모리에 존재하지 않는다면 서블릿 객체를 하나 생성하고 스레드에게 해당 작업을 위임한다.

해당 서블릿에서 요청에 대한 동적 페이지를 생성한 후 HttpServletResponse객체에 응답을 낸다. 응답이 끝나면 HttpServletRequest, HttpServletResponse 두 객체를 소멸시킨다.

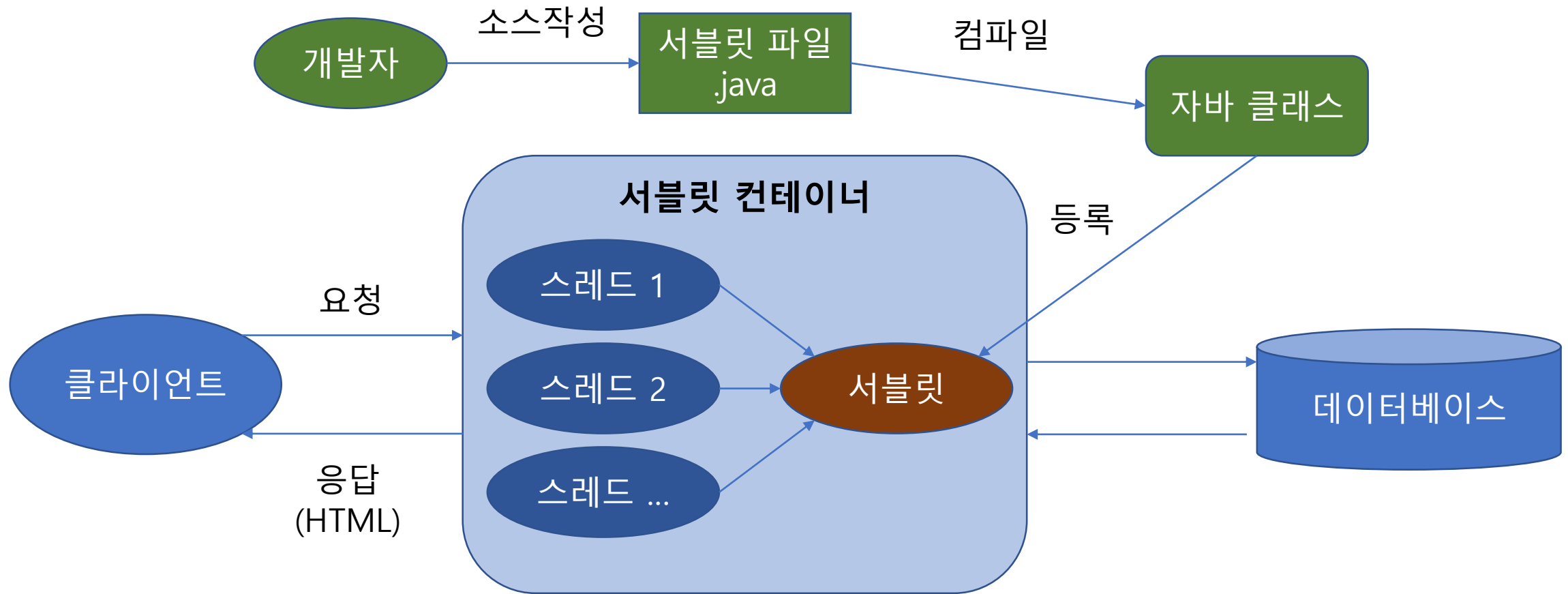


# Servlet 작동순서

서블릿은 JAVA를 기반으로 하기 때문에 서버 안의 JVM(멀티 스레드 지원)을 통해 요청을 스레드로 보낸다. 만약 새로운 요청이 들어왔을 때는 다른 스레드를 생성해서 요청을 처리한다. 스레드를 이용해서 요청을 처리하기 때문에, 서버 부하가 적게 발생한다. (속도가 빠르고, 효율적으로 서버를 운영할 수 있다)

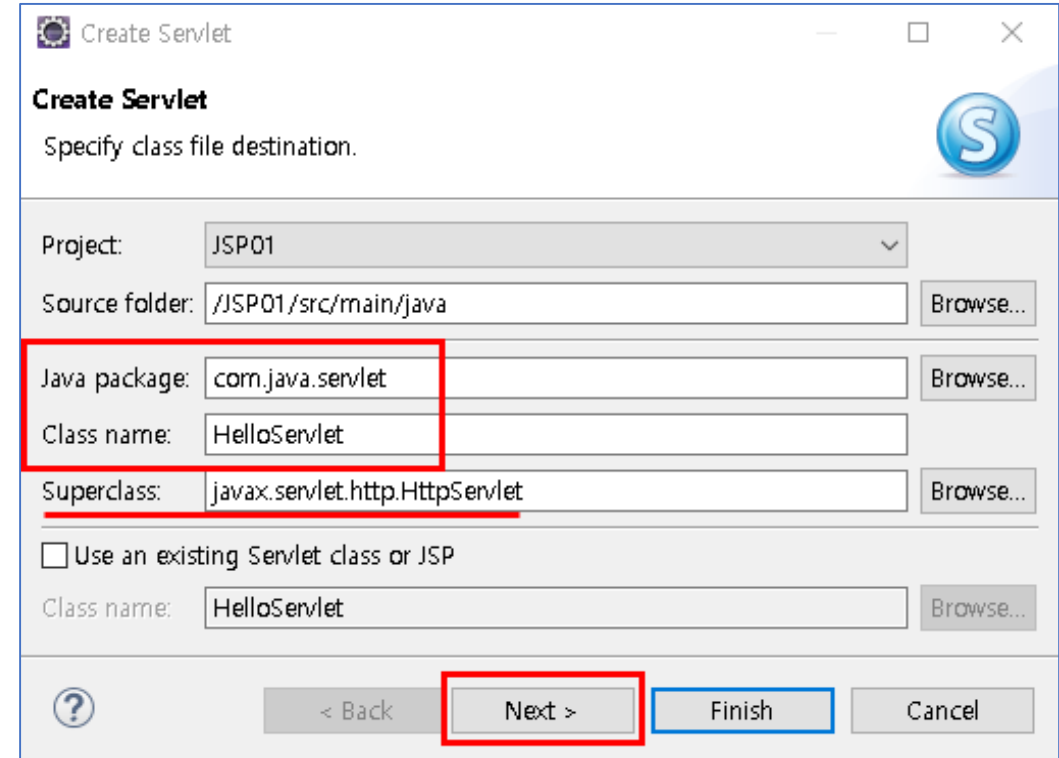
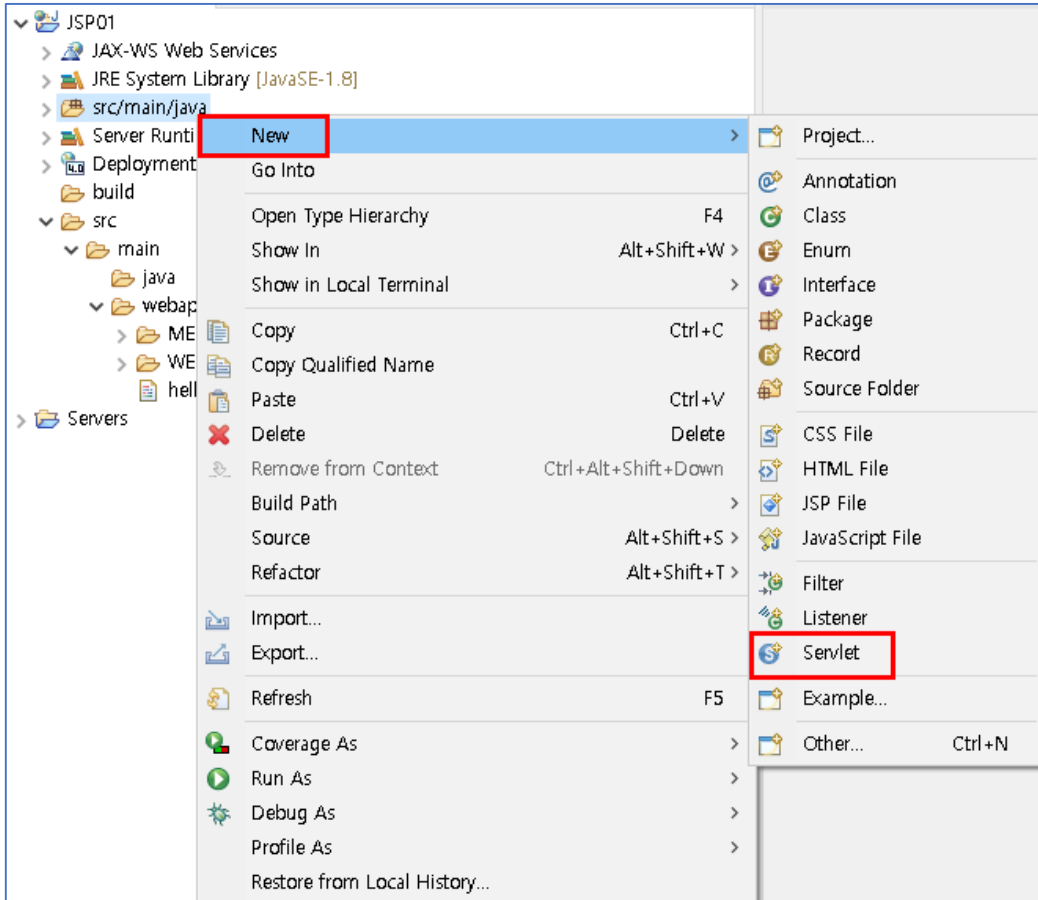


# Servlet 작동순서





# 서블릿 파일 만들기





# 서블릿 파일 만들기

Create Servlet

Create Servlet

Enter servlet deployment descriptor specific information.

Name: HelloServlet

Description:

Initialization parameters:

URL Mappings

Pattern: /Hello

URL mappings:

/HelloServlet

Asynchronous Support

Next >

Create Servlet

Create Servlet

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces:

Which method stubs would you like to create?

☒ Constructors from superclass

☒ Inherited abstract methods

☐ init ☐ destroy ☐ getServletConfig

☐ getServletInfo ☐ service ☒ doGet

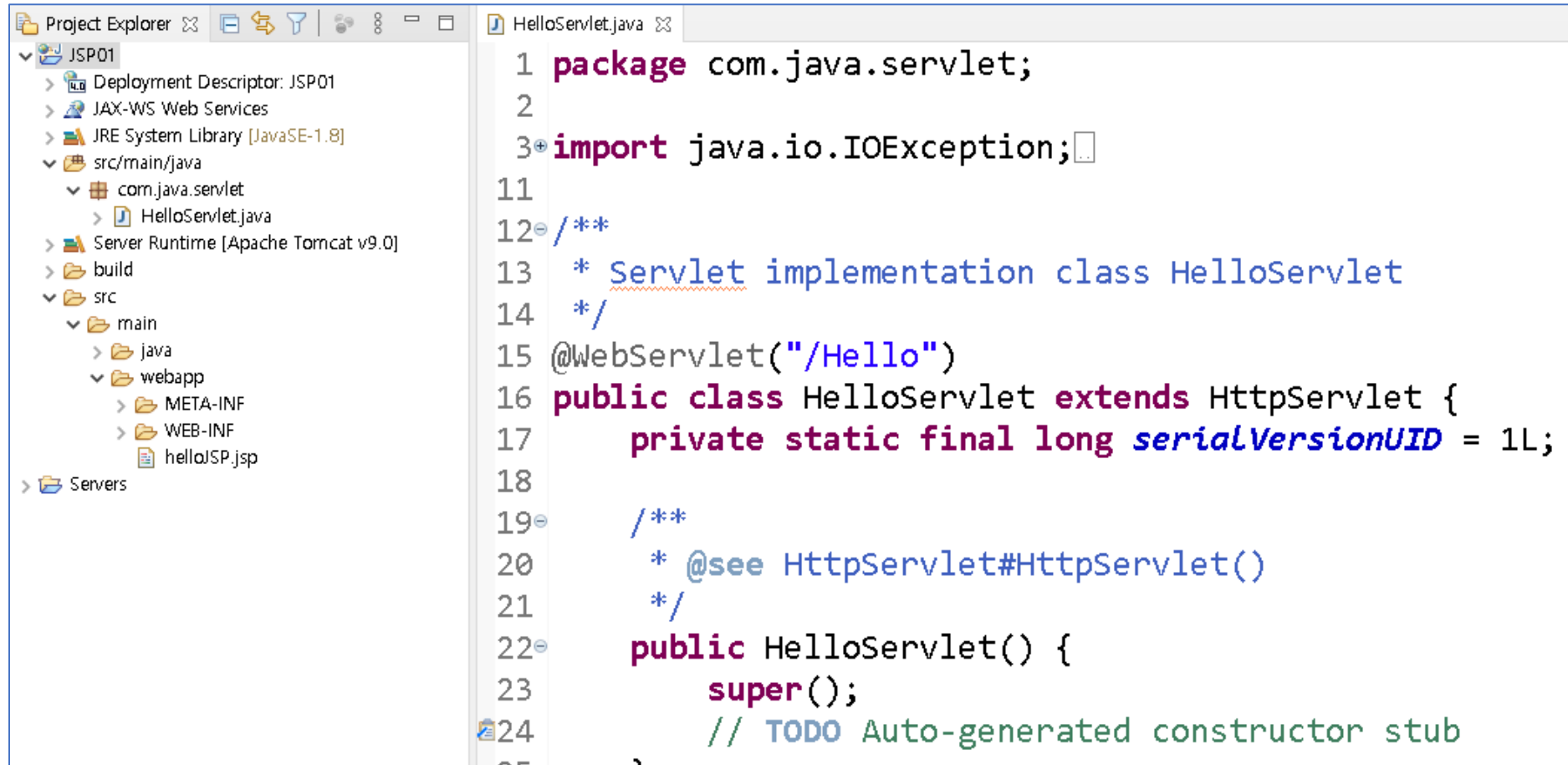
☒ doPost ☐ doPut ☐ doDelete

☐ doHead ☐ doOptions ☐ doTrace

Finish



# 서블릿 파일 만들기



```
1 package com.java.servlet;
2
3 import java.io.IOException;
4
5
6
7
8
9
10
11
12 /**
13  * Servlet implementation class HelloServlet
14  */
15 @WebServlet("/Hello")
16 public class HelloServlet extends HttpServlet {
17     private static final long serialVersionUID = 1L;
18
19     /**
20      * @see HttpServlet#HttpServlet()
21      */
22     public HelloServlet() {
23         super();
24         // TODO Auto-generated constructor stub
25     }
26 }
```





# Context Path

WAS(Web Application Server)에서 웹 어플리케이션을 구분하기 위한 path 이다.  
이클립스에서 프로젝트를 생성하면, 자동으로 server.xml에 추가된다.

url의 포트명 다음에 나온다.

http://localhost:8090/JSP01/hw

```
150
151
152 <!-- Access log processes all example.
153      Documentation at: /docs/config/valve.html
154      Note: The pattern used is equivalent to using pattern=...
155 <Valve className="org.apache.catalina.valves.AccessLogValve"
156
157 <Context docBase="JSP01" path="/JSP01" reloadable="true"
</Engine>
```



# Servlet Mapping

Servlet Mapping이란 특정 Servlet을 실행할 때, 해당하는 URL 패턴을 의미한다.  
(전체 경로를 URL 에 써주면 복잡하고 보안에도 취약하다.)

web.xml을 이용하는 방식과 어노테이션을 이용하는 방식이 있다.

```
<servlet>
  <servlet-name>helloServlet</servlet-name>
  <servlet-class>com.java.servlet.Servlet01</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>helloServlet</servlet-name>
  <url-pattern>/hw</url-pattern>
</servlet-mapping>
```

```
@WebServlet("/hw")
public class Servlet01
    private static fina
```



# Servlet 클래스



Servlet 클래스는 HttpServlet 클래스를 상속받는다.

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



# Servlet 클래스 활용

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
    response.setContentType("text/html; charset=UTF=8");  
  
    PrintWriter out = response.getWriter();  
  
    out.print("<!DOCTYPE html>");  
    out.print("<html>");  
    out.print("<head>");  
    out.print("<title>Servlet 시작!<title>");  
    out.print("</head>");  
    out.print("<body>");  
    out.print("<h1>Hello~~ Servlet</h1>");  
    out.print("</body>");  
    out.print("</html>");  
  
    out.close();  
  
}
```

요청처리 : request  
응답처리 : response

콘텐츠 형식의 설정 HTML의 경우 : "text/html"  
인코딩 설정 : "UTF-8"

웹 브라우저에 출력하기 위한 스트림  
PrintWriter out = response.getWriter();



# HTTP 패킷

클라이언트가 서버로 요청을 했을때, 보내는 데이터를 HTTP 패킷이라 표현한다.  
HTTP패킷의 구조는 크게 헤더 와 바디로 나뉘어진다.

헤더에는 HTTP 메서드 방식중 무엇을 썼는지, 클라이언트의 정보, 브라우저 정보, 접속할 URL 등등 과 같은 클라이언트 정보를 담는다.

바디는 보통 비어있다. 하지만, 특정 데이터를 담아서 보낼 수 있다.

이러한 웹 개념아래, ( GET / POST ) 방식을 통해서 서버에 요청을 할 수 있다.



# GET & POST 방식

GET 방식 : HTTP 패킷의 해더(Header)에 포함되어 서버에 요청한다.

URL뒤에 ? 를 사용하여 Parameter를 작성하게 되고 & 을 붙여 여러 개의 Parameter를 구분하게 된다. URL 값으로 정보가 전송되어 보안에 약하다.

POST 방식 : POST방식은 패킷의 BODY에다가 데이터를 넣어서 보낸다.

데이터 전송양에 길이 제한이 없으며 대용량 데이터를 보내는데 적합하다.

URL에 데이터가 노출되지 않아서 기본 보안은 되어있다.

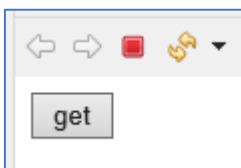
HTML 내 form 태그의 method속성에 따라 get이나 post 방식으로 요청할 수 있다.



# GET & POST 방식

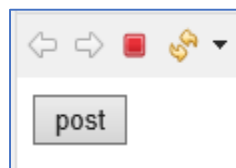
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    //response.getWriter().append("Served at: ").append(request.getContextPath());  
    System.out.println("doGet 메소드 입니다.");  
}  
  
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    //doGet(request, response);  
    System.out.println("doPost 메소드 입니다.");  
}
```

```
<form action="Servlet01" method="get">  
    <input type="submit" value="get">  
</form>
```



정보: 서버가 [778] 밀  
doGet 메소드 입니다.

```
<form action="Servlet01" method="post">  
    <input type="submit" value="post">  
</form>
```



정보: 서버가 [797] 밀  
doPost 메소드 입니다.



# request & response

클라이언트로부터 요청이 들어오면 WAS는 요청할 때 가지고 있는 정보를 `HttpServletRequest` 객체를 생성하여 저장한다. 또한 웹 브라우저에게 응답을 보낼 때 사용하기 위해 `HttpServletResponse` 객체를 생성하고, 생성된 두 객체를 서블릿에 전달한다.

HttpServletRequest	HttpServletResponse
http프로토콜의 request 정보를 서블릿에게 전달하기 위한 목적으로 사용한다.	WAS는 해당 클라이언트에게 응답을 보내기 위한 <code>HttpServletResponse</code> 객체를 생성하여 서블릿에게 전달한다.
헤더정보, 파라미터, 쿠키, URI, URL 등의 정보를 읽어 들이는 메소드를 가지고 있다.	서블릿은 해당 객체를 이용하여 Content type, 응답코드, 응답 메시지 등을 전송한다.



# Servlet Parameter

form 태그의 submit 버튼을 클릭하여 데이터를 서버로 전송하면 해당 Servlet 에서는 `HttpServletRequest` 객체를 이용해서 Parameter 값을 얻을 수 있다.

관련 메소드	설명
<code>String getParameter(name)</code>	파라미터 변수 <code>name</code> 에 저장된 변수값을 얻어내는 메소드로, 이때 변수의 값은 <code>String</code> 으로 리턴된다.
<code>String[] getParameterValues(name)</code>	파라미터 변수 <code>name</code> 에 저장된 모든 변수값을 얻어내는 메소드로, 이때 변수의 값은 <code>String</code> 배열로 리턴된다. <code>checkbox</code> 에서 주로 사용된다.
<code>Enumeration getParameterNames()</code>	요청에 의해 넘어오는 모든 파라미터 변수의 이름을 <code>java.util.Enumeration</code> 타입으로 리턴한다.



# Parameter 한글처리

Tomcat 서버의 기본 문자 처리방식은 IOS-8859-1 방식이다. 따라서 개발자가 별도의 한글 인코딩을 하지 않으면 한글이 깨져 보이는 현상이 있다.  
GET 방식과 POST 방식에 따라서 한글처리 방식에 차이가 있다.

```
-->  
<Connector URIEncoding="UTF-8"  
<!-- A "Connector" using the sh  
<!--
```

**GET 방식 요청**  
(server.xml 수정)

```
protected void doPost(HttpServletRequest request,  
  
    response.setCharacterEncoding("UTF-8");  
    request.setCharacterEncoding("UTF-8");
```

**POST 방식 요청**



# Servlet Parameter

```
<form action="Servlet04" method="post">

    이름 : <input type="text" name="name" size="10"> <br/>
    아이디 : <input type="text" name="id" size="10"> <br/>
    비밀번호 : <input type="password" name="pw" size="10"> <br/>

    취미 : <input type="checkbox" name="hobby" value="read">독서
    <input type="checkbox" name="hobby" value="walk">산책
    <input type="checkbox" name="hobby" value="swim">수영
    <input type="checkbox" name="hobby" value="cook">요리
    <input type="checkbox" name="hobby" value="run">달리기 <br/>

    전공 : <input type="radio" name="major" value="com">컴퓨터
    <input type="radio" name="major" value="math">수학
    <input type="radio" name="major" value="eng">영어
    <input type="radio" name="major" value="economy">경제 <br/>

    <select name="area">
        <option value="seoul">서울</option>
        <option value="daejeon">대전</option>
        <option value="daegu">대구</option>
        <option value="busan">부산</option>
    </select>
    <br/>

    <input type="submit" value="전송"><input type="reset" value="초기화">
</form>
```

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

String name = request.getParameter("name");
String id = request.getParameter("id");
String pw = request.getParameter("pw");

String[] hobbies = request.getParameterValues("hobby");
String major = request.getParameter("major");
String area = request.getParameter("area");

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

PrintWriter out = response.getWriter();

out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>form태그 알아보기</title>");
out.println("</head>");
out.println("<body>");
out.println("<h3>form태그에서 받은 데이터</h3>");
out.println("이름 : " + name + "<br/>");
out.println("아이디 : " + id + "<br/>");
out.println("비밀번호 : " + pw + "<br/>");
out.println("취미 : " + Arrays.toString(hobbies) + "<br/>");
out.println("전공 : " + major + "<br/>");
out.println("사는지역 : " + area + "<br/>");
out.println("</body>");
out.println("</html>");
```



# 초기화 파라미터

특정 Servlet이 생성될때 초기에 필요한 데이터들이 있다. (아이디 , 특정 정보 등)  
이러한 데이터들을 초기화 파라미터라고 한다. web.xml에 기술하고 Servlet 파일  
에서는 ServletConfig 객체로 접근 한다. 초기화 파라미터를 web.xml이 아닌  
Servlet 파일에 어노테이션으로 직접 기술하는 방법도 있다.

```
<init-param>
  <param-name>id</param-name>
  <param-value>java</param-value>
</init-param>

<init-param>
  <param-name>pw</param-name>
  <param-value>1234</param-value>
</init-param>
```

```
initParams = {@WebInitParam(name="id", value="java")}
```

# 데이터 공유

특정 데이터를 여러 Servlet에서 공유해야 할 경우가 있다. web.xml 에 데이터를 기술하고, Servlet에서는 ServletContext 객체로 접근하여 데이터를 얻을 수 있다.

```
<context-param>
  <param-name>id</param-name>
  <param-value>abcd</param-value>
</context-param>

<context-param>
  <param-name>pw</param-name>
  <param-value>1234</param-value>
</context-param>
```

```
String id = getServletContext().getInitParameter("id");
String pw = getServletContext().getInitParameter("pw");

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

web.xml 파일에  
parameter 기술



parameter 데이터 불러오기



# HttpSession

서블릿 HttpSession은 웹 애플리케이션에서 사용자의 세션 정보를 유지하고 관리하는 데 사용되는 인터페이스이다. 세션은 클라이언트(웹 브라우저)와 서버 간의 상태 정보를 유지하기 위해 사용된다.

HttpSession 객체는 웹 애플리케이션 내에서 고유한 세션 식별자를 가지고 있으며, 클라이언트가 애플리케이션과 상호작용하는 동안 세션 데이터를 저장하고 조회할 수 있는 메서드를 제공한다.

# HttpSession

메소드	설명
<code>getAttribute(String name)</code>	지정된 이름의 세션 속성 값을 가져온다.
<code>setAttribute(String name, Object value)</code>	지정된 이름으로 세션 속성 값을 설정한다.
<code>removeAttribute(String name)</code>	지정된 이름의 세션 속성 값을 제거한다.
<code>getId()</code>	세션 식별자를 반환한다.
<code>setMaxInactiveInterval(int interval)</code>	세션의 최대 유효 시간을 설정한다.
<code>getMaxInactiveInterval()</code>	세션의 최대 유효 시간을 반환한다.
<code>invalidate()</code>	세션을 무효화하고 세션과 관련된 모든 데이터를 제거한다.
<code>isNew()</code>	세션이 새로 생성되었는지 여부를 반환한다.



# Forward / Redirect

서블릿에서는 클라이언트의 요청을 다른 서블릿이나 JSP(JavaServer Pages)로 전달할 수 있는 두 가지 방법이 있다. 바로 Forward 와 Redirect 이다.

Forward는 서블릿 컨테이너(웹 서버) 내에서 클라이언트의 요청을 다른 서블릿이나 JSP로 전달하는 방식이다. Forward는 웹 브라우저에게는 다른 페이지로 이동한 것으로 보이지 않고, 클라이언트의 요청을 처리하는 동안에만 유효하다.

Redirect는 클라이언트에게 다른 URL로 이동하도록 요청하는 방식이다.

Redirect는 서버에서 클라이언트에게 특정 URL로 이동하도록 전달하고, 클라이언트는 새로운 URL로 새로운 요청을 보내는 방식이다.