

서블릿의 이해

목차

1. 서블릿 개요
2. 서블릿 구조와 생명주기
3. [기본실습] 서블릿 프로그래밍 : Hello World
4. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현
5. JSP와 서블릿의 관계

학습목표

- JSP를 실행할 때 컨테이너 내부에서 일어나는 과정을 이해한다.
- JSP의 기반이 되는 서블릿 구조를 이해한다.
- 서블릿의 소스코드를 이해하고 간단한 프로그램을 개발해본다.
- JSP와 서블릿의 관계를 이해한다.

01. 서블릿 개요

1. 서블릿(Servlet)이란?

- 서블릿은 자바 플랫폼에서 컴포넌트를 기반으로 하는 웹 애플리케이션 개발의 핵심 기술.
- JSP 는 서블릿 기반의 웹 프로그래밍 기술로 내부적으로 JSP는 서블릿으로 변환 되어 실행됨.
- 따라서 JSP를 보다 잘 이해하고 고급 웹 프로그래밍 개발을 위해서는 서블릿에 대한 이해가 필요함.

■ 서블릿의 장점

- ❶ 자바를 기반으로 하므로 자바 API를 모두 사용할 수 있다.
- ❷ 운영체제나 하드웨어에 영향을 받지 않으므로, 한 번 개발된 애플리케이션은 다양한 서버 환경에서도 실행할 수 있다.
- ❸ 웹 애플리케이션에서 효율적인 자료 공유 방법을 제공한다.
- ❹ 다양한 오픈소스 라이브러리와 개발도구를 활용할 수 있다.

01. 서블릿 개요

■ 웹 애플리케이션 개발에 서블릿 사용 시 이점

- ❶ MVC 패턴을 쉽게 적용할 수 있고 컨테이너와 밀접한 서버 프로그램을 구현할 수 있다.
- ❷ MVC 패턴을 적용할 때 콘텐츠와 비즈니스 로직을 분리할 수 있으며 컨트롤러와 뷰가 역할을 분담함으로써, 웹 디자이너와 개발자 간에 작업을 원활하게 할 수 있다.
- ❸ 리스너 및 필터 서블릿 등 고급 프로그래밍 기법을 통해 더욱 효과적인 웹 애플리케이션을 설계할 수 있다.

최근에는 스프링이나 스트러츠 같은 오픈소스 프레임워크가 주목받고 있으며, 이들 프레임워크의 많은 부분이 내부적으로 서블릿 기술을 이용하고 있으므로 프레임워크 기반의 웹 애플리케이션을 개발하려면 서블릿을 잘 배워두는 것이 좋다.

01. 서블릿 개요

2. 서블릿과 서블릿 컨테이너

- 서블릿 컨테이너는 서블릿을 실행하기 위한 서버 소프트웨어를 말하는 것으로 JSP나 서블릿으로 만들어진 웹 프로그램을 개발하고 실행하기 위한 환경임.
- 아파치 톰캣이 대표적임.

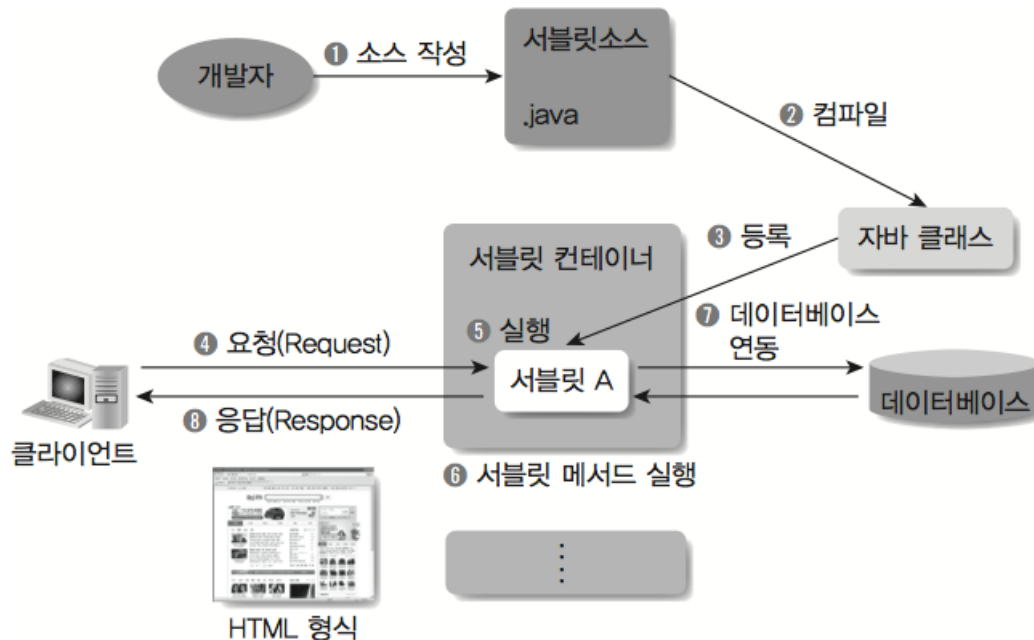
[표 4-1] 웹 서버와 서블릿 컨테이너

구분	웹 서버	서블릿 컨테이너
사용목적	웹 서비스를 제공하기 위해 필요한 서버 기반의 소프트웨어다.	서블릿으로 개발된 자바 프로그램을 실행하고 처리하기 위한 서버 기반의 소프트웨어다.
처리 콘텐츠	HTML, CSS, 자바스크립트, 이미지 파일 등이다.	서블릿 클래스다.
실행 방법	콘텐츠가 위치한 URL 요청에 의해 실행하며 요청할 때마다 매번 디스크에서 읽어 처리한다.	서블릿 클래스 정보에 따라 서버에 매핑된 URL 정보에 따라 실행하며 컨테이너에 적재된 상태에서 처리한다.
JSP 실행	자체로 처리할 수 없다. 서블릿 컨테이너로 처리를 넘긴다.	JSP 자체로 처리할 수 있다.
특징	웹 서비스 제공을 위한 다양한 설정을 제공하기 때문에 서버를 유연하게 운영하려면 웹 서버를 사용해야 한다.	컨테이너에 따라 기본적인 웹 서버 기능을 내장하고 있으나 고급 설정이나 성능이 떨어지기 때문에 웹 서버와 병행해서 사용할 것을 권장한다.

01. 서블릿 개요

3. 서블릿 동작 과정

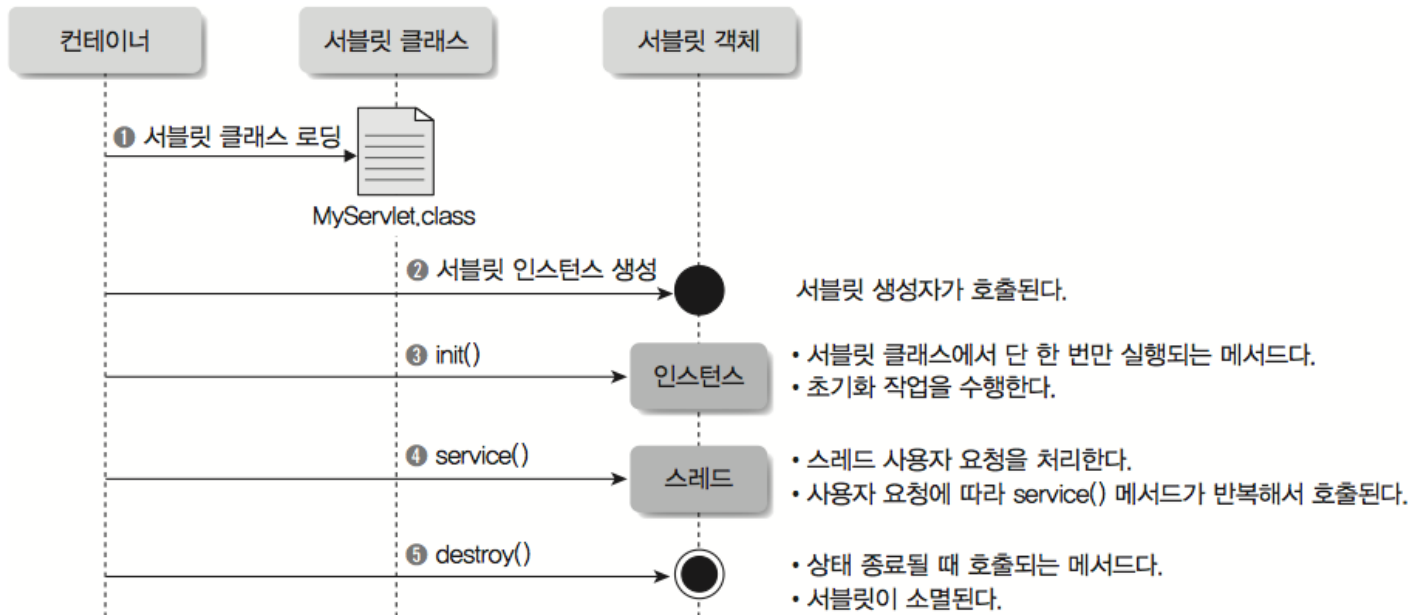
- 서블릿은 개발자가 소스 작성 후 컴파일 과정을 거쳐 컨테이너에 배치(deploy)하게 되면 컨테이너에 의해 실행되어 관리된다.
- 이후 사용자 요청에 따라 스레드 단위로 실행되면서 데이터베이스 연동 등 필요한 작업을 수행하고 처리 결과를 사용자에게 HTML 형식으로 전달하는 구조로 동작한다.



[그림 4-1] 서블릿 개발과 실행 과정

01. 서블릿 개요

- 서블릿은 일반적인 애플리케이션처럼 버튼을 누르면 시작되고 처리를 마치면 종료되는 구조가 아님.
- 서버에서 컨테이너에 의해 실행 되면서 생명주기를 가지며 특정 이벤트와 상태가 존재하는 구조.
- 서블릿 개발은 해당 생명주기 메서드를 오버라이딩하거나 doGet(), doPost()와 같은 사용자 요청 처리 메서드를 구현하는 것임.
- init()은 서블릿 실행시 한번만 실행되는 메서드이고 service() 메서드는 사용자 요청시 매번 호출되는 메서드이다. destroy() 는 서블릿 종료 시 실행되는 메서드이다.



[그림 4-2] 서블릿의 동작 과정

02. 서블릿 구조와 생명주기

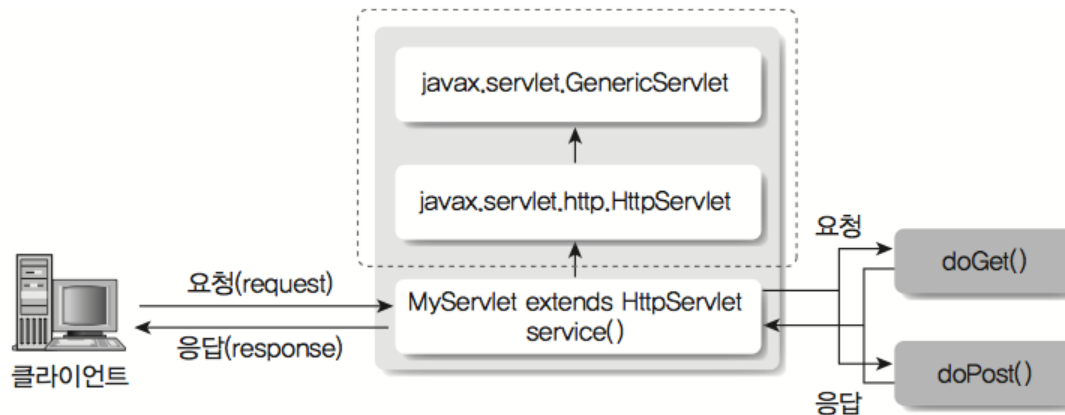
1. 서블릿 구조와 API

- JSP와 달리 서블릿은 일반적인 자바 클래스 구조를 가진다.
- 이는 서블릿이 일반 자바 소스의 구조라 는 의미로 컴파일 과정이 필요함을 의미
- 또한 서블릿은 컨테이너에 의해 실행되므로 개발자가 임의로 프로그램 하는 것이 아니라 특정 클래스를 상속 받아야만 구현할 수 있는 구조임.
- 따라서 서블릿 프로그램을 하려면 서블릿 클래스의 상관 관계나 API의 기본 구조를 이해해야 한다.
- API(Application Programming Interface)는 특정 클래스를 다른 프로그램에서 사용하기 위해 필요한 정보를 규격화 해놓은 것을 말함.
- 일반적으로 서블릿은 `java.servlet.HttpServlet` 클래스를 상속해서 구현 함.

02. 서블릿 구조와 생명주기

■ javax.servlet.http.HttpServlet 동작 구조

- GenericServlet 에 비해 HTTP 프로토콜 지원이 포함되어 일반적인 웹 프로그램에 적합.
- HttpServlet 도 javax.servlet.GenericServlet 을 상속받고 있음.
- 사용자 요청에 따라 GET, POST 방식으로 구별해 처리하지만 경우에 따라서는 구분없이 처리하기도 함.
- <http://www.joby.co.kr/index.html> 이라는 URL 요청은 HTTP 프로토콜에서는 GET /index.html 과 같이 서버에 전달됨.
- HTTP 프로토콜에는 GET, POST, PUT, HEAD, DELETE, OPTIONS, TRACE 와 같은 요청이 정의되어 있으며 서블릿에도 각각 doGet() , doPost()와 같은 대응 메서드가 존재함.



[그림 4-4] javax.servlet.http.HttpServlet을 상속받은 서블릿 동작 구조

02. 서블릿 구조와 생명주기

■ GET 방식

- 서버에 있는 정보를 클라이언트로 가져오기 위한 방법이다. 예를 들어 HTML, 이미지 등을 웹 브라우저에서 보기 위한 요청.
- 서버에는 최대 240Byte까지 데이터를 전달할 수 있다.
- QUERY_STRING 환경변수를 통해서 서버로 전달되는데, 다음 형식을 따른다.
 - `http://www.xxx.co.kr/servlet/login?id=hj&name=hong`
- '?' 이후의 값들은 서버에서 QUERY_STRING을 통해 전달된다. '속성=값' 형태로 사용해야 하며 '&'는 여러 속성 값을 전달할 때 연결해주는 문자열이다.
- URL이 노출되기 때문에 보안에 문제가 생길 수 있다.

■ POST 방식

- 서버로 정보를 올리기 위해 설계된 방법이다. 예를 들어 HTML 폼에 입력한 내용을 서버에 전달하기 위한 요청.
- 서버에 전달 할 수 있는 데이터 크기에는 제한이 없다.
- URL에는 매개변수가 표시되지 않는다.

02. 서블릿 구조와 생명주기

2. 서블릿 생명주기

■ 서블릿 초기화 : **init()** 메서드

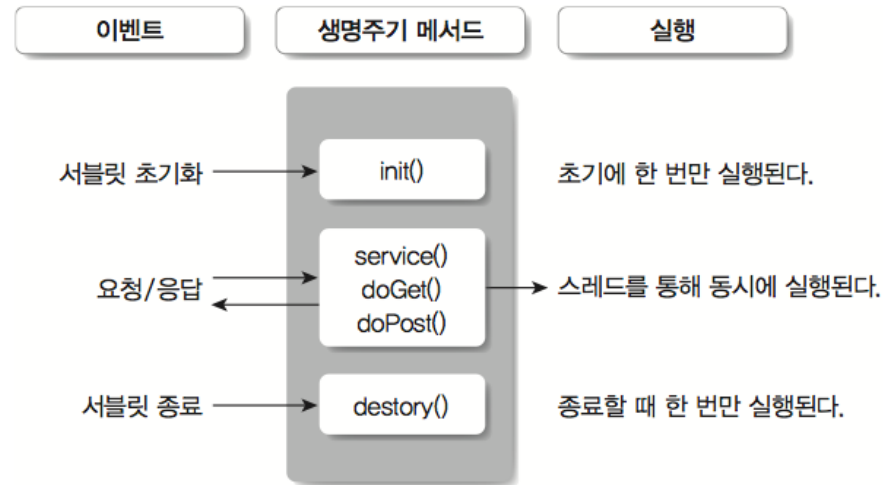
- 서블릿 실행시 호출되는 메서드로 초기에 한 번만 실행된다. 공통적으로 필요한 작업 등 수행

■ 요청/응답 : **service()** 메서드

- 사용자 요청에 따라 스레드로 실행되는 메서드로 각각 service() 메서드를 통해 doGet() 혹은 doPost() 메서드가 호출된다.
- 파라미터인 HttpServletRequest 와 HttpServletResponse 를 통해 사용자 요청을 처리한다.

■ 서블릿 종료 : **destroy()** 메서드

- 컨테이너로부터 서블릿 종료 요청이 있을 때 호출되는 메서드.
- init()와 마찬가지로 한 번만 실행되며, 서블릿이 종료되면서 정리할 작업이 있다면 destroy() 를 오버라이딩해서 구현함.



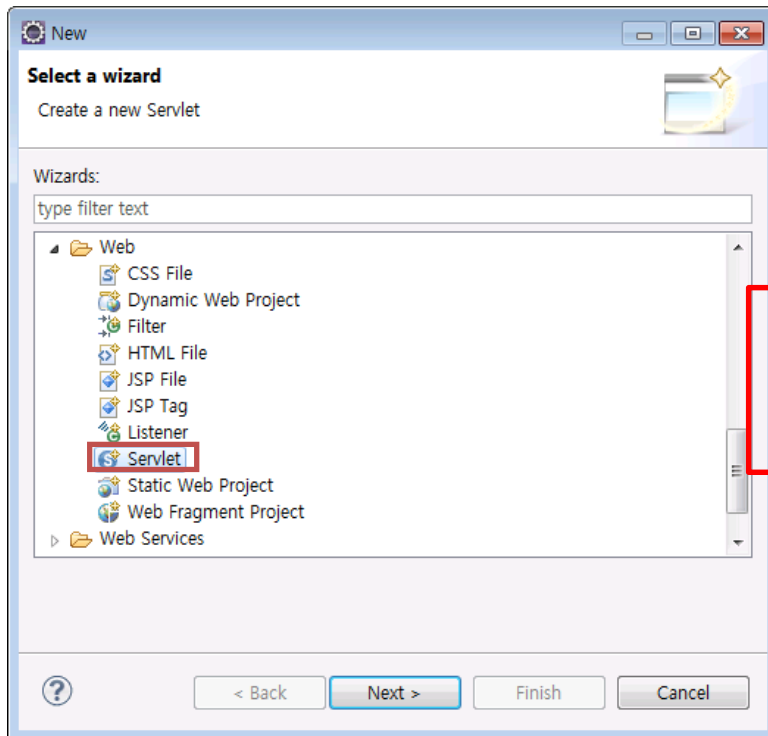
[그림 4-5] 서블릿 생명주기

03. [기본실습] 서블릿 프로그래밍 : Hello World

1. 서블릿 생성

■ 서블릿 마법사 시작

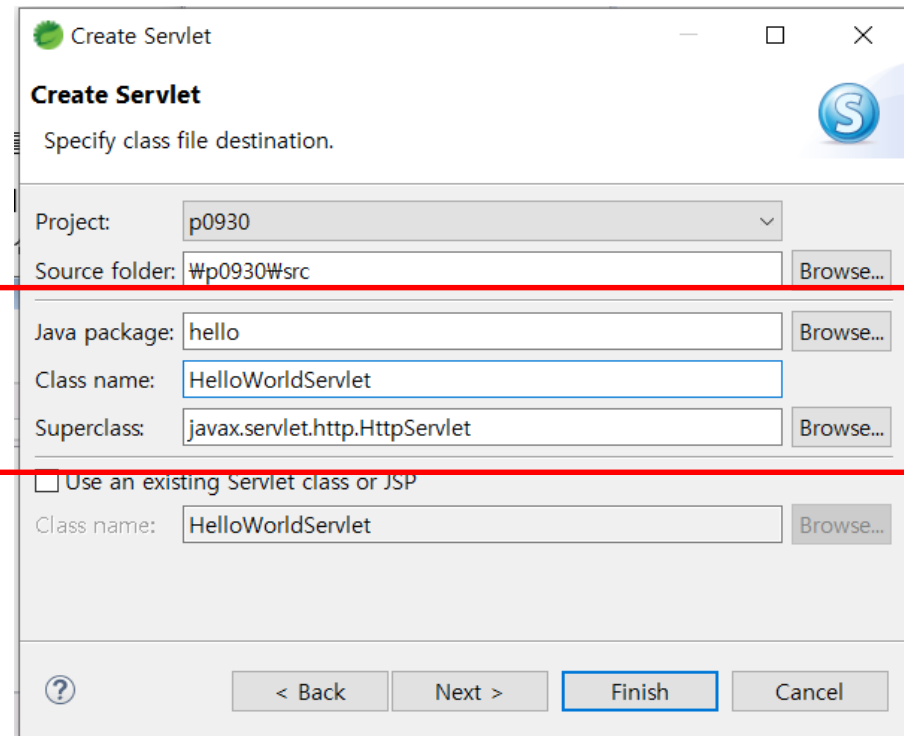
- [WebContent] → [servlet] 폴더를 만든다.
- <마우스 오른쪽> 버튼을 클릭해 [New] → [Other] → [Web] → [Servlet]을 선택한다.



[그림 4-6] 서블릿 선택

■ 기본 정보 입력

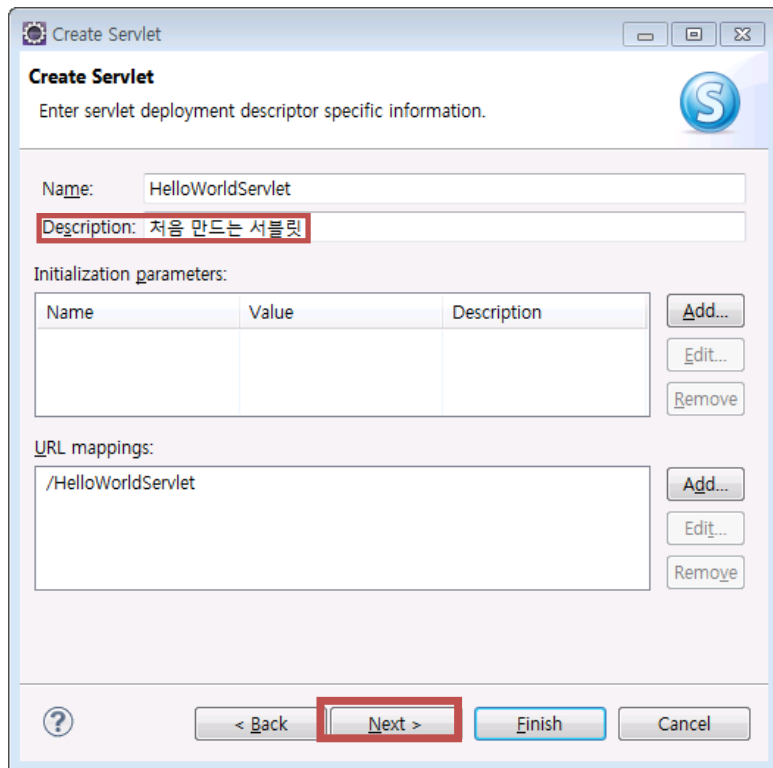
- 패키지 : hello
- 클래스 : HelloWorldServlet



[그림 4-7] 기본 정보 입력

03. [기본실습] 서블릿 프로그래밍 : Hello World

■ Web.xml 설정



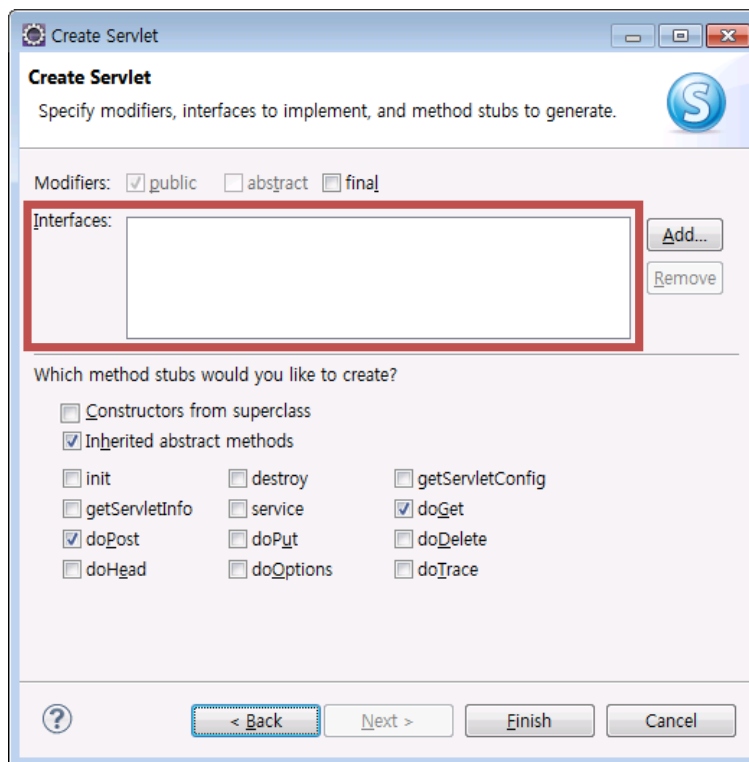
The 'Create Servlet' dialog box is shown with the following configuration:

- Name:** HelloWorldServlet
- Description:** 처음 만드는 서블릿 (highlighted with a red box)
- Initialization parameters:** A table with columns Name, Value, and Description. It is currently empty.
- URL mappings:** /HelloWorldServlet
- Buttons:** At the bottom, the '< Back' and 'Next >' buttons are highlighted with a red box.

[그림 4-8] web.xml 설정

■ 클래스 설정

- 1 'Interfaces'에 등록되어 있는 내용을 삭제한다.
[Constructors from superclass] 항목은 체크하지 않는다.



The 'Create Servlet' dialog box is shown with the following configuration:

- Modifiers:** ☒ public, ☐ abstract, ☐ final
- Interfaces:** A text area for listing interfaces, highlighted with a red box.
- 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
- Buttons:** At the bottom, the '< Back' and 'Next >' buttons are highlighted with a red box.

[그림 4-9] 클래스 설정

03. [기본실습] 서블릿 프로그래밍 : Hello World

② <Finish> 버튼을 눌러 생성된 서블릿 코드를 확인한다

```
1 HelloWorldServlet.java
2
3 package hello;
4
5 import java.io.IOException;
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 /**
13  * Servlet implementation class HelloWorldServlet
14  */
15 @WebServlet(description = "처음 만드는 서블릿", urlPatterns = { "/HelloWorldServlet" })
16 public class HelloWorldServlet extends HttpServlet {
17     private static final long serialVersionUID = 1L;
18
19     /**
20      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
21      */
22     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
23         // TODO Auto-generated method stub
24         response.getWriter().append("Served at: ").append(request.getContextPath());
25     }
26
27     /**
28      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
29      */
30     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
31         // TODO Auto-generated method stub
32         doGet(request, response);
33     }
34 }
```

03. [기본실습] 서블릿 프로그래밍 : Hello World

2. 소스코드 분석

- 서블릿 패키지 import : 서블릿 클래스 API 사용을 위해 필요한 기본 패키지

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

■ 서블릿 클래스 선언

- @WebServlet : 컨테이너에 서블릿임을 알리는 애너테이션
- urlPatterns={"/HelloWorldServlet"} : 서블릿 실행을 위한 요청 URL
- javax.servlet.http.HttpServlet 클래스를 상속해서 필요한 라이프사이클 메서드를 구현

```
// 서블릿 관련 애너테이션 설정
@WebServlet(description = "첫째 만드는 서블릿", urlPatterns = { "/HelloWorldServlet" })
public class HelloWorldServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * 기본 생성자
     */
    public HelloWorldServlet() {
    }
```


03. [기본실습] 서블릿 프로그래밍 : Hello World

■ 구현부 작성 : doGet() 메서드

- 클라이언트로 전달하게 될 콘텐츠 타입과 캐릭터셋 지정
- 브라우저 출력을 위한 출력 스트림 객체(java.io.PrintWriter) 가져옴

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

    // 콘텐츠 타입 선언 및 한글 설정
    response.setContentType("text/html;charset=UTF-8");

    // 웹브라우저 출력을 위한 PrintWriter 객체 확보
    PrintWriter out = response.getWriter();
    // HTML 형식으로 브라우저 출력 콘텐츠 작성
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Hello World Servlet</TITLE></HEAD>");
    out.println("<BODY><H2>Hello World Servlet : 헬로월드</H2></BODY>");
    out.println("</HTML>");
}
```

■ doGet() 메서드와 doPost() 메서드의 구현

- 요청 방식에 따른 메서드 구현, 여기서는 doPost()에서 doGet()을 호출하도록 해서 GET, POST 모두 doGet()에서 처리하는 것으로 함.

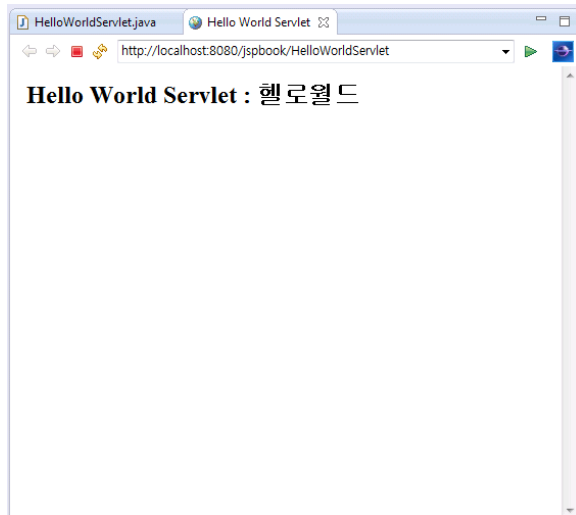
```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // Post에서 별다른 처리 없이 doGet으로 포워딩
    doGet(request, response);
}
```

03. [기본실습] 서블릿 프로그래밍 : Hello World

■ [실습] 서블릿을 이용한 헬로월드 프로그램(HelloWorldServlet.java)

3. 서블릿 실행

- HelloWorld.java 소스를 선택하고 <마우스 오른쪽> 버튼을 클릭해 [Run] → [Run on Server]를 선택한다.
- 별도의 브라우저를 통해 접속할 경우에는 다음 URL로 실행한다.
 - <http://localhost:8080/Hello/HelloWorldServlet>



[그림 4-13] 실행 결과 확인

04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

1. HttpServlet 클래스의 개요

■ doGet 메서드와 doPost 메서드

- HttpServletRequest와 HttpServletResponse 매개변수는 서블릿과 클라이언트 사이를 연결해주는 중요한 객체들로, HttpServlet 클래스의 특징이 된다.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
public void doPost(HttpServletRequest request, HttpServletResponse response)
```

2. HttpServlet 클래스를 이용하여 프로그래밍하기

■ 프로그램 개요

- CalcServlet.java : 서블릿에서 계산 로직과 화면 처리를 하는 버전
- CalcServlet2.java : 클래스를 통해서 계산 로직을 처리하면서 화면 처리에 중점을 둔 서블릿 버전

04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

■ 사용자 인터페이스의 구현

- 계산기 사용자 인터페이스(calc.html)

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5 <TITLE>ch04 : 계산기</TITLE>
6 </HEAD>
7
8 <BODY>
9 <CENTER>
10 <H3>계산기</H3>
11 <HR>
12 <!-- action 의 서블릿 요청 경로에 주의 해야함. 서블릿 클래스에 선언된 요청과 같아야 함 -->
13 <form name=form1 action=/p0930/CalcServlet method=post>
14 <INPUT TYPE="text" NAME="num1" width=200 size="5">
15 <SELECT NAME="operator">
16 <option selected>+</option>
17 <option>-</option>
18 <option>*</option>
19 <option>/</option>
20 </SELECT>
21 <INPUT TYPE="text" NAME="num2" width=200 size="5">
22 <input type="submit" value="계산" name="B1"> <input type="button" value="다시입력" name="B2">
23 </form>
24 </CENTER>
25 </BODY>
26 </HTML>
```

계산기

<input type="text"/>	+ ▼	<input type="text"/>	계산	다시입력
----------------------	-----	----------------------	----	------

04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

■ 계산기 서블릿의 구현 ①

- 계산 기능이 통합된 서블릿(CalcServlet.java)

```
1 package hello;
2
3 // 패키지 import
4 import java.io.*;
5 import javax.servlet.*;
6 import javax.servlet.annotation.WebServlet;
7 import javax.servlet.http.*;
8
9 /**
10  * File : CalcServlet.java
11  * Desc : 계산기 서블릿
12  * @author
13  */
14 @WebServlet(description = "Calc1 서블릿", urlPatterns = { "/CalcServlet" })
15 public class CalcServlet extends HttpServlet {
16
17     private static final long serialVersionUID = 1L;
18
19     // GET 요청을 처리하기 위한 메서드
20     public void doGet(HttpServletRequest request, HttpServletResponse response)
21         throws ServletException, IOException {
22         // doPost()로 포워딩.
23         doPost(request, response);
24     }
25 }
```

04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

■ 계산기 서블릿의 구현 ①

- 계산 기능이 통합된 서블릿(CalcServlet.java)

```
// POST 요청을 처리하기 위한 메서드
// doGet()에서도 호출하고 있기 때문에 모든 요청은 doPost()에서 처리되는 구조이다.
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // 변수 선언
    int num1, num2;
    int result;
    String op;

    // 클라이언트 응답시 전달될 콘텐츠에 대한 타입 설정과 캐릭터셋 지정
    response.setContentType("text/html; charset=UTF-8");

    // 클라이언트 응답을 위한 출력 스트림 확보
    PrintWriter out = response.getWriter();

    // HTML 폼을 통해 전달된 num1, num2 파라미터 값을 변수에 할당한다.
    // 이때 getParameter() 메서드는 문자열을 리턴하므로 숫자형 데이터의 경우 Integer.parseInt() 를 통해 int로 변환 한다.
    num1 = Integer.parseInt(request.getParameter("num1"));
    num2 = Integer.parseInt(request.getParameter("num2"));
    op = request.getParameter("operator");
    // calc() 메서드 호출로 결과를 받아 온다.
    result = calc(num1, num2, op);
}
```

04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

■ 계산기 서블릿의 구현 ①

- 계산 기능이 통합된 서블릿(CalcServlet.java)

```
// 출력 스트림을 통해 화면을 구성 한다.  
out.println("<HTML>");  
out.println("<HEAD><TITLE>계산기</TITLE></HEAD>");  
out.println("<BODY><center>");  
out.println("<H2>계산결과</H2>");  
out.println("<HR>");  
out.println(num1+" "+op+" "+num2+" = "+result);  
out.println("</BODY></HTML>");  
}
```

```
//실제 계산 기능을 수행하는 메서드  
public int calc(int num1, int num2, String op) {  
    int result = 0;  
  
    if(op.equals("+")) {  
        result = num1 + num2;  
    }  
    else if(op.equals("-")) {  
        result = num1 - num2;  
    }  
    else if(op.equals("*")) {  
        result = num1 * num2;  
    }  
    else if(op.equals("/")) {  
        result = num1 / num2;  
    }  
    return result;  
}  
}
```

04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

■ 계산기 서블릿의 구현 ②

- 계산 기능이 별도로 분리된 클래스(CalcServlet2.java)
- 계산기 서블릿 소스에서 계산에 필요한 메서드를 별도의 클래스로 분리하는 방법.

```
1 package hello;
2
3 // 패키지 import
4 import java.io.*;
5
6 import javax.servlet.*;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.*;
9
10 /**
11  * File : CalcServlet2.java
12  * Desc : 계산기 서블릿2
13  * @author
14  */
15 @WebServlet(description = "Calc2 서블릿", urlPatterns = { "/CalcServlet2" })
16 public class CalcServlet2 extends HttpServlet {
17
18     private static final long serialVersionUID = 1L;
19
20     // GET 요청을 처리하기 위한 메서드
21     public void doGet(HttpServletRequest req, HttpServletResponse res)
22         throws ServletException, IOException {
23         // doPost()로 포워딩.
24         doPost(req, res);
25     }
```


04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

■ 계산기 서블릿의 구현 ②

- 계산 기능이 별도로 분리된 클래스(CalcServlet2.java)
 - 계산기 서블릿 소스에서 계산에 필요한 메서드를 별도의 클래스로 분리하는 방법.

```
// POST 요청을 처리하기 위한 메서드
// doGet()에서도 호출하고 있기 때문에 모든 요청은 doPost()에서 처리되는 구조이다.
public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
    // 변수선언
    int num1, num2;
    int result;
    String op;

    // 클라이언트 응답시 전달될 컨텐트에 대한 타입 설정과 캐릭터셋 지정
    res.setContentType("text/html; charset=UTF-8");

    // 클라이언트 응답을 위한 출력 스트림 확보
    PrintWriter out = res.getWriter();

    // HTML 문을 통해 전달된 num1, num2 파라미터 값을 변수에 할당한다.
    // 이때 getParameter() 메서드는 문자열을 리턴하므로 숫자형 데이터의 경우 Integer.parseInt() 를 통해 int로 변환 한다.
    num1 = Integer.parseInt(req.getParameter("num1"));
    num2 = Integer.parseInt(req.getParameter("num2"));
    op = req.getParameter("operator");

    // Calc 클래스 인스턴스 생성 후 getResult 메서드 호출로 결과 받음
    Calc calc = new Calc(num1, num2, op);
    result = calc.getResult();
}
```

04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

■ 계산기 서블릿의 구현 ②

- 계산 기능이 별도로 분리된 클래스(CalcServlet2.java)
 - 계산기 서블릿 소스에서 계산에 필요한 메서드를 별도의 클래스로 분리하는 방법.

```
// 출력 스트림을 통한 화면 구성
out.println("<HTML>");
out.println("<HEAD><TITLE>계산기</TITLE></HEAD>");
out.println("<BODY><center>");
out.println("<H2>계산결과</H2>");
out.println("<HR>");
out.println(num1+" "+op+" "+num2+" = "+result);
out.println("</BODY></HTML>");
}
}
```

04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

- 독립된 계산 기능 구현 클래스(Calc.java)
 - 기존 calc() 메서드 내용을 별도 클래스로 구성한다.
 - [src] → [hello] 패키지를 선택한 상태에서 [New] → [Class]를 실행하고 Calc라는 클래스를 생성한다.

```
1 package hello;
2 /**
3  * File : Calc.java
4  * Desc : 계산기 클래스
5  */
6 public class Calc {
7     // 결과값 저장을 위한 변수
8     int result = 0;
9     // 생성자에서 계산을 바로 수행함
10    public Calc(int num1, int num2, String op) {
11        if(op.equals("+")) {
12            result = num1 + num2;
13        }
14        else if(op.equals("-")) {
15            result = num1 - num2;
16        }
17        else if(op.equals("*")) {
18            result = num1 * num2;
19        }
20        else if(op.equals("/")) {
21            result = num1 / num2;
22        }
23    }
24    // 계산 결과를 리턴하는 메서드
25    public int getResult() {
26        return result;
27    }
28 }
```

04. [응용실습] 서블릿 프로그래밍 : 계산기 서블릿 구현

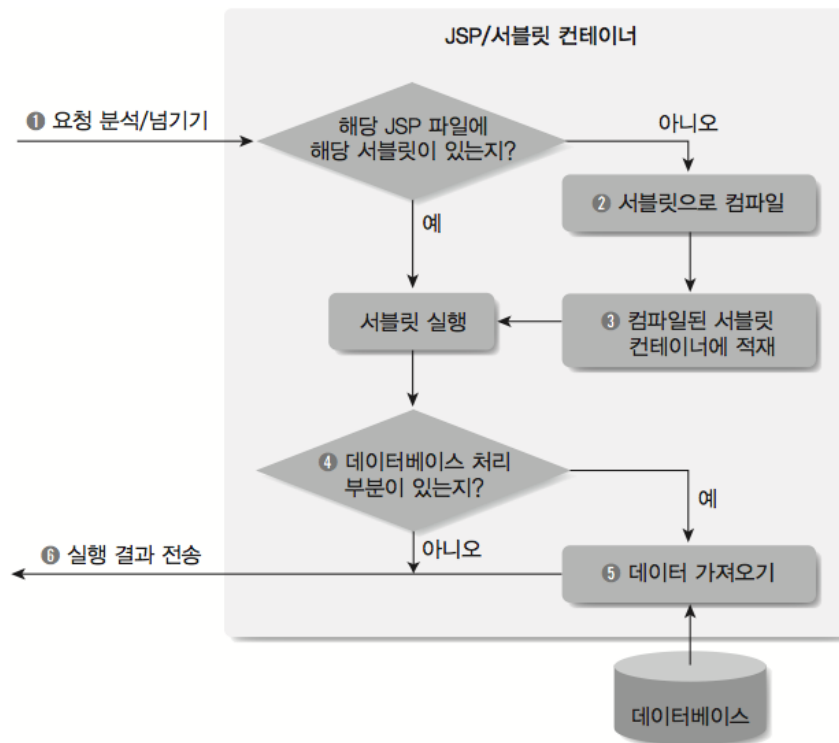
■ Calc.html 수정

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5 <TITLE>ch04 : 계산기</TITLE>
6 </HEAD>
7
8 <BODY>
9 <CENTER>
10 <H3>계산기</H3>
11 <HR>
12 <!-- action 의 서블릿 요청 경로에 주의 해야함. 서블릿 클래스에 선언된 요청과 같아야 함 -->
13 <form name=form1 action=/p0930/CalcServlet method=post>
14 <INPUT TYPE="text" NAME="num1" width=200 size="5">
15 <SELECT NAME="operator">
16 <option selected>+</option>
17 <option>-</option>
18 <option>*</option>
19 <option>/</option>
20 </SELECT>
21 <INPUT TYPE="text" NAME="num2" width=200 size="5">
22 <input type="submit" value="계산" name="B1"> <input type="reset" value="다시입력" name="B2">
23 </form>
24 </CENTER>
25 </BODY>
26 </HTML>
```

05. JSP와 서블릿 관계

■ JSP 파일의 서블릿 변환 처리 과정

- `http://localhost:8080/jspbook/ch03/HelloWorld.jsp` 가 요청될 경우 컨테이너(톰캣)은 해당 JSP 파일에 대한 서블릿이 있는지 확인하고 없다면 `HelloWorld_jsp.java` 파일을 생성함.
- `HelloWorld_jsp.java`는 다시 `HelloWorld_jsp.class` 로 컴파일되어 실행되고 톰캣에 의해 관리됨.
- JSP에서의 사용자 구현 코드는 `_jspService()`에 위치하며, jsp파일 요청시 실행되게 됨.

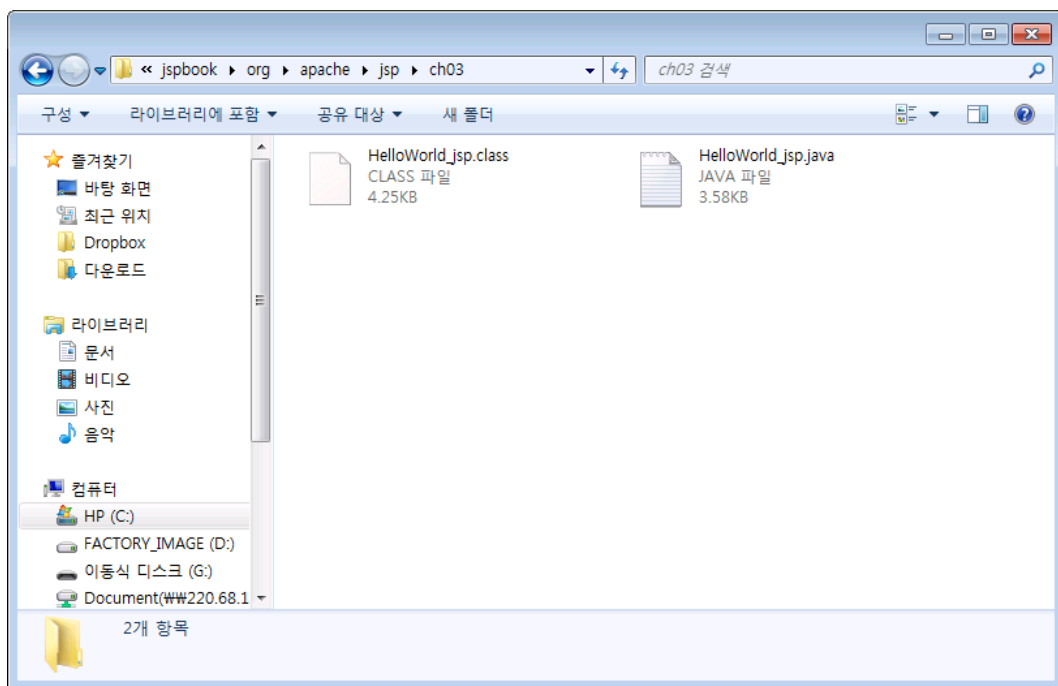


[그림 4-18] JSP 서블릿 컴파일과 처리 과정

05. JSP와 서블릿 관계

■ 변환된 HelloWorld_jsp.java 소스 분석

- 변환된 파일 위치
 - c:\dev\workspace\metadata\plugins\org.eclipse.wst.server.core\Wtmp0\work\Catalina\localhost\jspbook\org\apache\jsp\ch03
 - 폴더내 파일을 삭제한 후 이클립스에서 jsp를 실행하면 다시 파일이 생성되는 것을 볼 수 있음.



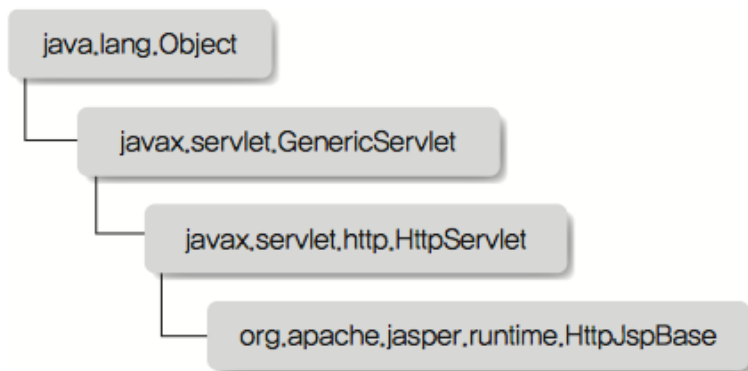
[그림 4-19] 컨테이너에 의해 자바 소스로 변환된 JSP 파일

05. JSP와 서블릿 관계

■ 패키지 및 클래스 선언 부분

- 서블릿과 jsp 처리 관련 클래스 import

```
package org.apache.jsp.ch03;  
import javax.servlet.*;  
import javax.servlet.http.*;  
import javax.servlet.jsp.*;  
public class HelloWorld_jsp extends  
org.apache.jasper.runtime.HttpJspBase implements  
org.apache.jasper.runtime.JspSourceDependent {  
}
```



[그림 4-19] HttpJspBase 클래스 구조

05. JSP와 서블릿 관계

■ `jspService()` 메서드 선언 부분

```
public void _jspService(HttpServletRequest request, HttpServletResponse response) throws java.io.IOException, ServletException {
```

- request, response를 파라미터로 받고 있다. request와 response는 JSP 내장 객체중 하나로 사용자 요청을 처리하는데 필요한 기본적인 요소들이다. 6장에서 자세히 살펴본다.

■ JSP 내장객체 선언 및 초기화 부분

- `jspServ`이 외에 `pageContext`, `application`, `config`, `session`, `out` 도 모두 내장객체가 된다. 내장객체는 서블릿으로 변환된 JSP에서 별도의 선언 없이 접근 가능한 객체들로 `jsp`파일에서 스크립트릿 등을 이용해 바로 사용할 수 있는 객체들을 말한다. 각각의 객체들의 고유 기능은 API로 제공된다.

```
try{  
    _jspxFactory = JspFactory.getDefaultFactory();  
    response.setContentType("text/html;charset=euc-kr");  
    pageContext = _jspxFactory.getPageContext(this,  
request,  
    response, null, true, 8192, true);  
    application = pageContext.getServletContext();  
    config = pageContext.getServletConfig();  
    session = pageContext.getSession();  
    out = pageContext.getOut();  
    _jspx_out = out;  
    ...  
}
```


05. JSP와 서블릿 관계

■ 개발자가 작성한 JSP 코드 부분

- JSP 파일에서 개발자가 작성한 코드는 모두 `out.write()` 형태로 처리된다.

```
out.write("<BODY>");
out.write("<H2>Hello World : 헬로월드");
out.write("</H2>WrWn오늘의 날짜와 시간은 : ");
out.write(new java.util.Date());
out.write("WrWn");
out.write("</BODY>WrWn");
out.write("</HTML>");
...
```

- 이러한 코드 구조를 알아두면 JSP가 서블릿 기술에 기반한다는 것을 명확하게 알 수 있으며 JSP의 구조나 동작 원리 등을 이해하는데 도움이 된다.
- 또한 디버깅이나 오류 추적에도 유용하게 활용할 수 있다.