

서블릿의 기초

강사 : 강병준

서블릿

1. 서블릿은 JSP 표준이 나오기 전에 만들어 진 표준으로 자바에서 웹 애플리케이션을 개발할 수 있도록 하기 위해 만들어 졌으며 자바 클래스를 웹에서 호출 및 실행 할 수 있도록 한 표준
2. javax.servlet.http.HttpServlet 클래스로부터 상속받아서 작성
3. 위의 클래스는 톰캣의 servlet-api.jar 에 포함되어 있습니다.
4. 작성과정
 - 1) 서블릿 규칙에 따라 자바 코드 생성
 - 2) 작성한 코드를 컴파일해서 웹 프로젝트의 WEB-INF의 classes 폴더에 복사
 - 3) 경우에 따라서 web.xml 파일에서 서블릿을 주소와 매핑
 - 4) 웹 컨테이너 재실행
5. 서블릿 요청 처리
 - 1) 요청 방식에 따라 doGet이나 doPost 메서드를 재정의해서 처리
 - 2) service 메서드를 재정의해서 사용할 수 있는데 이 메서드는 get 방식이나 post 방식 상관없이 호출되며 이 메서드가 호출되면 doGet이나 doPost 메서드는 호출되지 않습니다.

서블릿

1. 서블릿에서 요청을 처리하기 위해 오버라이딩 한 메서드는 request 객체를 이용해서 웹 브라우저의 요청 정보를 읽어 오던가 아니면 response를 이용해서 응답을 전송할 수 있습니다.
2. 응답을 전송하고자 하는 경우는 response 객체의 `setContentType()` 메서드를 이용해서 타입과 인코딩 방식을 지정해 주어야 합니다.
3. 웹 브라우저에 데이터를 전송하려면 `getWriter()`를 호출해서 문자열 데이터를 출력할 수 있는 `PrintWriter`를 가져오고 `print()`나 `println()`을 이용해서 전송하면 됩니다.

서블릿

새로운 Dynamic Web Project를 생성해서 src에 패키지를 생성하고
HelloServlet 라는 서블릿을 생성하고 실행

```
@WebServlet("/HelloServlet")
public class HelloServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>Hello Servlet</h1>");
        out.println("</body></html>");
        out.close();
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
    }
}
```

서블릿

1. 이클립스 5.0이전 버전에서는 web.xml 파일을 이용해서 매핑을 했지만 지금은 annotation의 등장으로 주소와의 매핑을 코드 안에서 가능한데 클래스 정의 상단에 @WebServlet(주소 또는 urlPatterns="패턴")의 형태로 가능
2. web.xml(웹 프로젝트의 설정 파일)에서의 매핑

```
<servlet>  
    <servlet-name>서블릿이름</servlet-name>  
    <servlet-class>클래스이름</servlet-class>  
</servlet>  
  
<servlet-mapping>  
    <servlet-name>서블릿이름</servlet-name>  
    <url-pattern>/호출url</url-pattern>  
</servlet-mapping>
```
3. <servlet-mapping> 엘리먼트 안에는 전체 URL이 아니라, 웹 서버의 도메인 이름, 포트 번호, 웹 어플리케이션 디렉터리 이름을 제외한 나머지 부분만 기재해야 합니다.
4. /* -> 무조건 수행
5. /aaa/* -> aaa 이면 무조건 수행
6. *.jsp -> jsp 확장자인 경우 무조건 수행

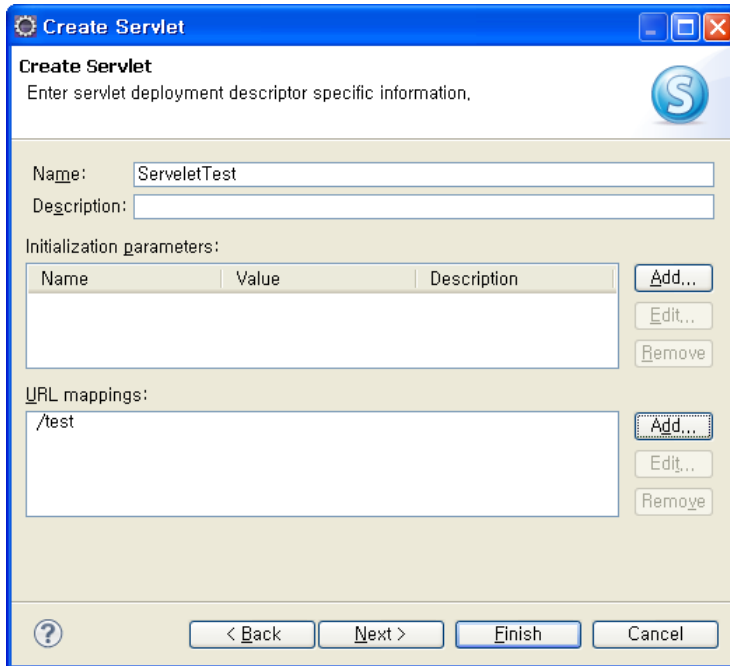
서블릿

1. 매개변수(parameter- 폼 안에서 name을 가진 객체, 또는 URL의 ? 다음의 데이터) 처리 방법
 - 1) 단일 항목인 경우: request 객체의 `getParameter("매개변수이름")`으로 처리
 - 2) 배열인 경우: request 객체의 `getParameterValues("매개변수이름")`으로 처리
2. 요청 시 응답
 - 1) 요청 시 응답은 response 객체를 이용
 - 2) 출력을 하고자 하는 경우라면 response 객체의 `getWriter()`를 호출해서 `PrintWriter` 객체를 받은 후 `print` 또는 `println` 메서드를 이용해서 html 코드를 리턴(`print`는 이어지는 문장을 계속 작성하고자 하는 경우 사용하고 `println`은 하나의 문장을 작성하기 위해서 사용)
3. 특정 페이지로 이동
 - 1) request 공유
`RequestDispatcher 변수 = request.getRequestDispatcher("이동할 페이지");`
`변수.forward(request, response);`
 - 2) request 공유하지 않고 이동
`response.sendRedirect("이동할 페이지")`

서블릿

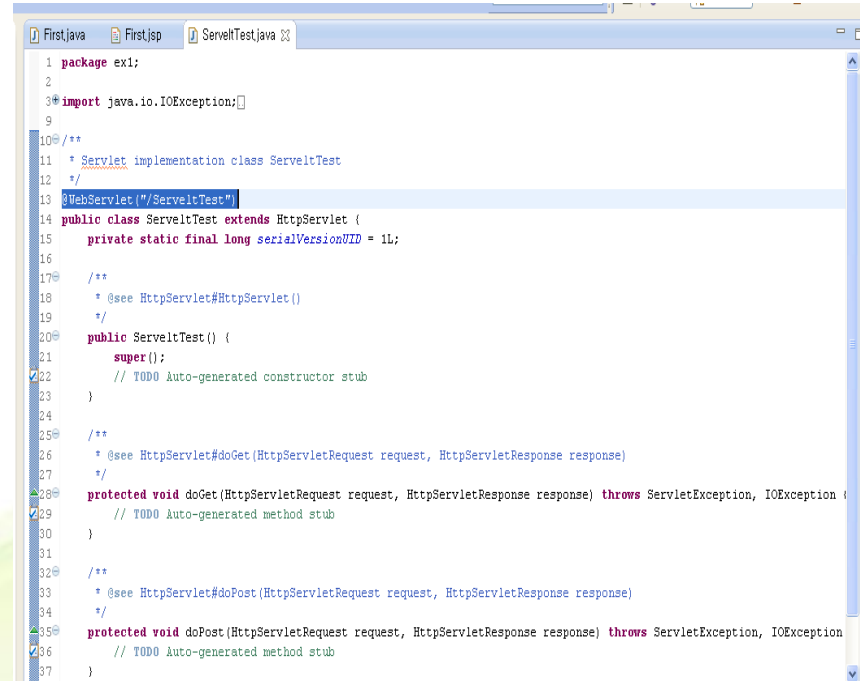
❖ API

<https://docs.oracle.com/javaee/7/api/>



The 'Create Servlet' dialog box is shown. It has a title bar with a gear icon and the text 'Create Servlet'. Below the title bar, it says 'Create Servlet' and 'Enter servlet deployment descriptor specific information.' There is a blue 'S' icon in the top right corner. The dialog contains several input fields and buttons:

- Name:** A text field containing 'ServletTest'.
- Description:** An empty text field.
- Initialization parameters:** A table with columns 'Name', 'Value', and 'Description'. There are 'Add...', 'Edit...', and 'Remove' buttons to the right of the table.
- URL mappings:** A text field containing '/test'. There are 'Add...', 'Edit...', and 'Remove' buttons to the right of the field.
- At the bottom, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.



The 'ServletTest.java' code editor is shown. It displays the following code:

```
1 package ex1;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 /**
11  * Servlet implementation class ServletTest
12  */
13 @WebServlet("/ServletTest")
14 public class ServletTest extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public ServletTest() {
21         super();
22         // TODO Auto-generated constructor stub
23     }
24
25     /**
26      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
27      */
28     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
29         // TODO Auto-generated method stub
30     }
31
32     /**
33      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
34      */
35     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
36         // TODO Auto-generated method stub
37     }
38 }
```

서블릿

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>now</servlet-name>
    <servlet-class>test.TimePrint</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>now</servlet-name>
    <url-pattern>/aaa/*</url-pattern>
  </servlet-mapping>

</web-app>
```


서블릿

서블릿 호출 방법

Get 방식 - 주소에 매개변수를 붙여서 호출하는 방식

- 1) 주소와 매개변수를 붙여서 주소 표시줄에 입력하는 방법(?로 구분)
- 2) a 태그를 이용해서 페이지를 요청하는 경우
- 3) 자바 스크립트를 이용해서 요청하는 경우
- 4) 폼에서 명시적으로 GET 방식으로 요청하는 경우
- 5) 매개변수의 데이터는 255자 이내이며 보안성이 없음
- 6) 폼에서 사용하면 처리가 지연되는 경우 재요청

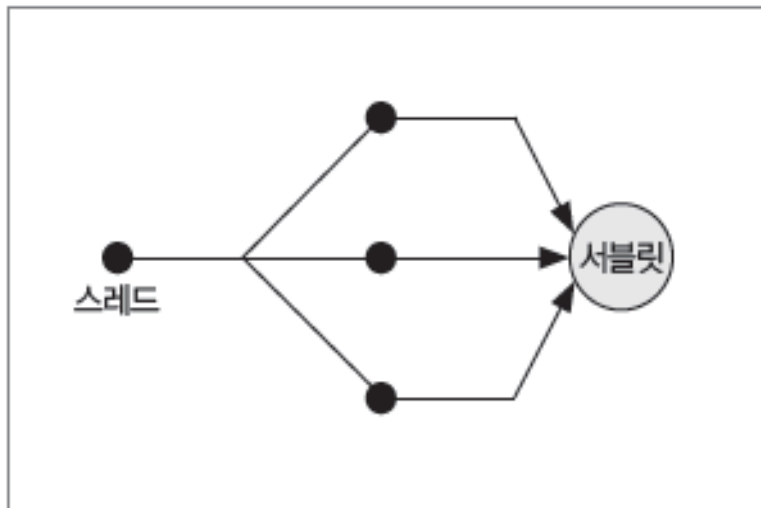
Post 방식 - 매개변수를 본문에 포함시켜 전송하는 방식

- 1) 폼에서 명시적으로 POST 방식으로 요청
- 2) 데이터의 크기에 제한이 없으며 URL에 표시가 되지 않으므로 보안성이 우수

서블릿

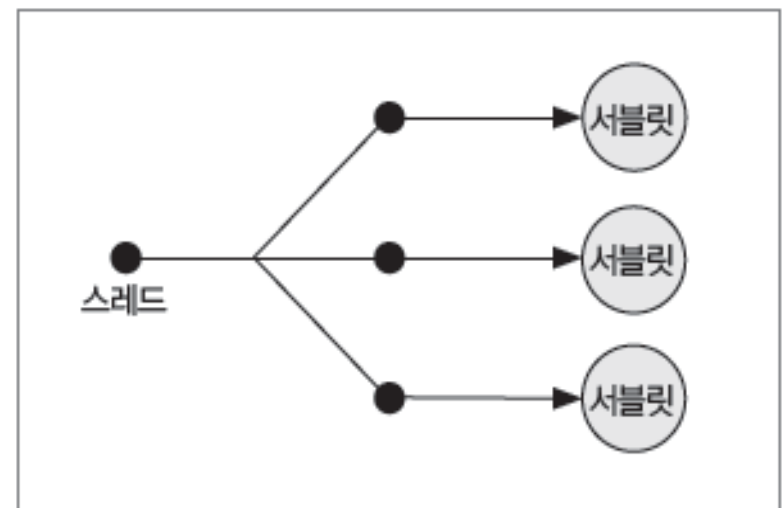
멀티스레드 모델의 장점: 필요한 서블릿의 수가 적기 때문에 서블릿을 만들기 위해 필요한 시스템 자원과 서블릿이 차지하는 메모리를 절약할 수 있다. 단점: 여러 스레드가 동시에 한 서블릿을 사용하기 때문에 데이터 공유 문제에 신경을 써야 한다.

웹 컨테이너



(a) 멀티-스레드 모델(Multi-Thread Model)

웹 컨테이너



(b) 싱글-스레드 모델(Single-Thread Model)

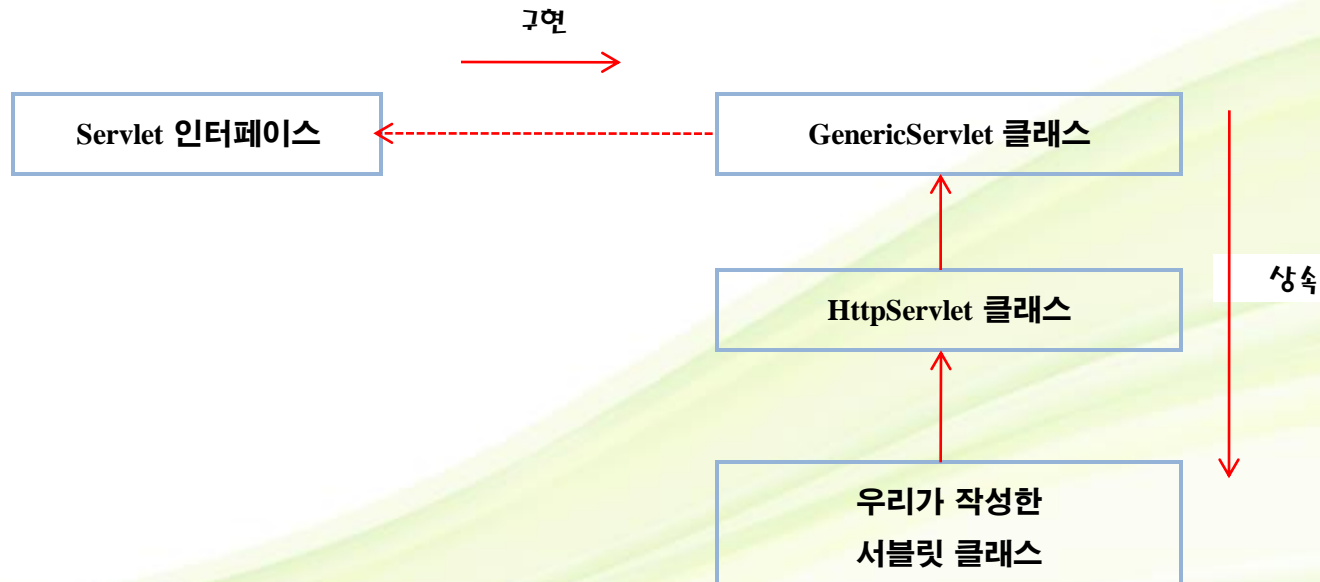
멀티-스레드 모델과 싱글-스레드 모델

- 싱글-스레드 모델에서는 데이터 공유 문제를 걱정할 필요가 없지만 시스템 자원과 메모리가 더 많이 소모된다.

서블릿 클래스의 작성, 컴파일, 설치, 등록

서블릿 클래스의 작성을 위한 준비

- 서블릿 클래스를 작성할 때 지켜야 할 규칙 세 가지
 - 서블릿 클래스는 javax.servlet.http.HttpServlet 클래스를 상속하도록 만들어야 한다
 - doGet 또는 doPost 메서드 안에 웹 브라우저로부터 요청이 왔을 때 해야 할 일을 기술해야 한다
 - HTML 문서는 doGet, doPost 메서드의 두 번째 파라미터를 이용해서 출력해야 한다



서블릿 클래스의 상속/구현 관계

서블릿 클래스의 작성, 컴파일, 설치, 등록

```
;
@WebServlet("/AdderServlet") // servlet와 serlet-mapping 대체
public class AdderServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String str1 = request.getParameter("num1");
        String str2 = request.getParameter("num2");
        int num1 = Integer.parseInt(str1);
        int num2 = Integer.parseInt(str2);
        int total = num1 + num2;
        response.setContentType("text/html;charset=euc-kr");
        PrintWriter out = response.getWriter();
        out.println("<html><body> <h1>두수의 합</h1>");
        out.println(num1 + " + " + num2 + " = " + total);
        out.println("</body></html>");
        out.close();
    }
}
```

서블릿 클래스의 작성, 컴파일, 설치, 등록

- doGet 메서드의 골격을 만든 다음에는 안에 내용을 채워 넣는다.

```
public class HundredServlet extends HttpServlet {  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        int total = 0;  
        for (int cnt = 1; cnt < 101; cnt++)  
            total += cnt;  
    }  
}
```

1부터 100까지의 합을 구하는 명령문

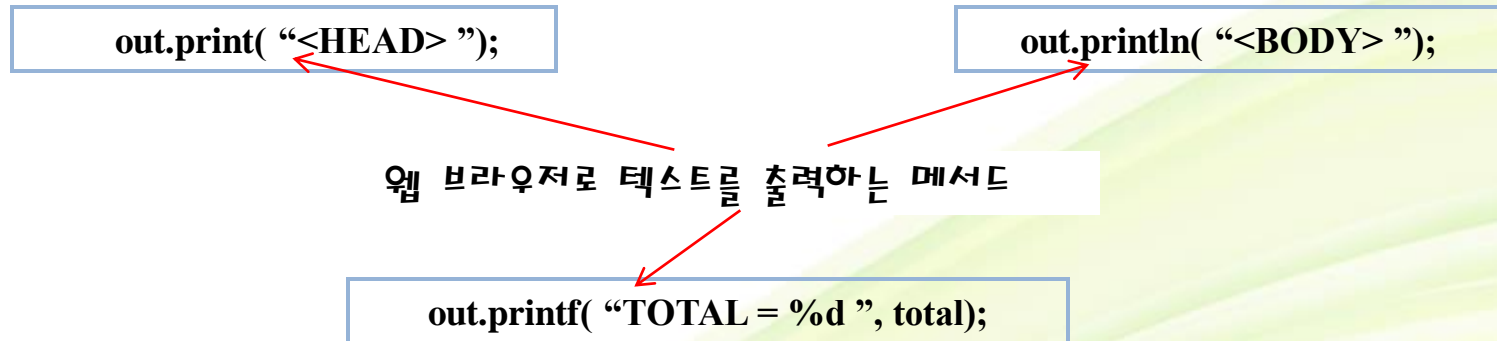
- 실행 결과를 출력하는 코드는 doGet 메서드의 두 번째 파라미터를 이용해서 작성한다.
- 두 번째 파라미터는 javax.servlet.http.HttpServletResponse 인터페이스 타입이며, 여기에 getWriter라는 메서드를 호출해서 PrintWriter 객체를 구한다.

```
PrintWriter out = response.getWriter();
```

PrintWriter 객체를 리턴하는 메서드

서블릿 클래스의 작성, 컴파일, 설치, 등록

- `PrintWriter`는 본래 자바 프로그램에서 파일로 텍스트를 출력할 때 사용하는 `java.io` 패키지의 `PrintWriter` 클래스이다.
- `Response.getWriter`메서드가 리턴하는 `PrintWriter` 객체는 파일이 아니라 웹 브라우저로 데이터를 출력한다.



서블릿 클래스의 작성, 컴파일, 설치, 등록

- 서블릿 클래스가 완성 되었으면, 코드에서 사용한 여러 가지 클래스와 인터페이스를 가져오는 import 문을 추가한다.

1부터 100까지 더하는 서블릿 클래스

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class HundredServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int total = 0;
        for (int cnt = 1; cnt < 101; cnt++)
            total += cnt;
        PrintWriter out = response.getWriter();
        out.println( "<HTML> ");
        out.println( "<HEAD><TITLE>Hundred Servlet</TITLE></HEAD> ");
        out.println( "<BODY> ");
        out.printf( "1 + 2 + 3 + ... + 100 = %d ", total);
        out.println( "</BODY> ");
        out.println( "</HTML> ");
    }
}
```

서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 한글 HTML 문서를 출력하는 서블릿 클래스

- 한글이 포함된 HTML 문서를 출력하려면 doGet, doPost 메서드의 두 번째 파라미터인 HttpServletResponse 타입의 파라미터에 대해 다음과 같은 메서드를 호출해야 한다.

```
response.setContentType( "text/html;charset=euc-kr " );
```

이 문서의 내용은 HTML 문법으로 작성된 텍스트이고

euc-kr 문자셋(한글 코드)로 인코딩되어 있음

- 이 명령문은 HTML을 출력하는 print, println, printf 메서드 호출문보다 앞에 와야 할 뿐만 아니라, response.getWriter 메서드 호출문보다도 먼저 와야 한다.

서블릿 클래스의 작성, 컴파일, 설치, 등록

- 한글을 포함한 HTML 문서를 출력하는 서블릿 클래스는 다음과 같다.

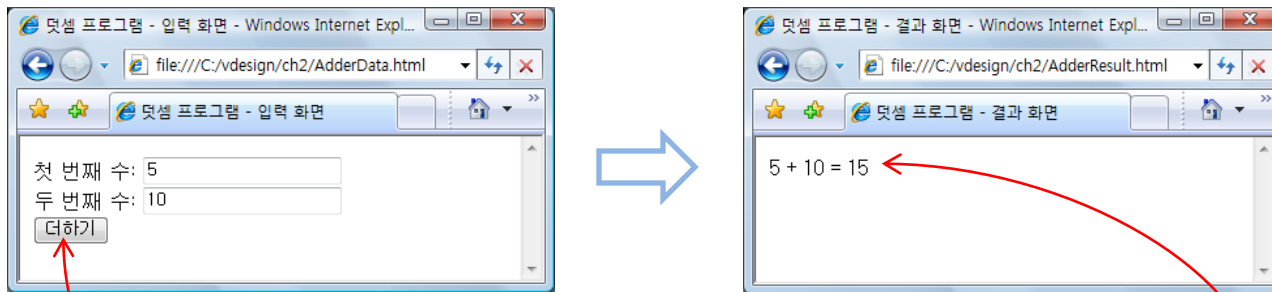
1부터 100까지 더하는 서블릿 클래스

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class HundredServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        int total = 0;
        for (int cnt = 1; cnt < 101; cnt++)
            total += cnt;
        response.setContentType( "text/html;charset=euc-kr ");
        PrintWriter out = response.getWriter();
        out.println( "<HTML> ");
        out.println( "<HEAD><TITLE>1부터 100까지 더하는 서블릿</TITLE></HEAD> ");
        out.println( "<BODY> ");
        out.printf( "1부터 100까지의 합은 = %d " , total);
        out.println( "</BODY> ");
        out.println( "</HTML> ");
    }
}
```

웹 브라우저로부터 데이터 입력받기

❖ 웹 브라우저로부터 데이터를 입력받는 서블릿 클래스

- 왼쪽 웹 페이지를 통해 두 수를 입력받은 후 그 둘을 합한 결과를 오른쪽 웹 페이지를 통해 보여주는 웹 애플리케이션이다.



[그림 2-21] 두 수의 합을 구하는 웹 애플리케이션의 화면 설계

① 두 수를 입력하고
더하기 버튼을 누르면

② 두 수의 합을 보여주는
웹 페이지가 나타난다.

- 둘 이상의 웹 페이지로 구성되는 웹 애플리케이션을 개발할 때는 먼저 화면 설계를 하고 다음에 각 화면의 URL을 정하고, 코딩 작업에 들어가는 것이 좋다.

웹 브라우저로부터 데이터 입력받기

두 개의 수를 입력받는 HTML 문서

```
<script type="text/javascript">
  function chk() {
    if (!frm.num1.value) {alert("숫자를 입력하세요");
      frm.num1.focus();return false;}
    if (isNaN(frm.num1.value)) {alert("문자입니다 다시");
      frm.num1.focus(); frm.num1.value=""; return false;}
    if (!frm.num2.value) {alert("입력하라니까 ?");
      frm.num2.focus();return false;}
    if (isNaN(frm.num2.value)) {alert("바보아냐 숫자냐 !");
      frm.num2.focus(); frm.num2.value=""; return false;}
    return true;
  }
</script> </head> <body>
<h1>두수 입력</h1>
<form action="Add" name="frm" onsubmit="return chk()" method="post">
  첫 번째 수 : <input type="text" name="num1"> <p>
  두 번째 수 : <input type="text" name="num2"> <p>
  <input type="submit" value="완료">
</form>
```

웹 브라우저로부터 데이터 입력받기

- 오른쪽 화면을 구현하는 서블릿 클래스는 을 통해 입력된 두 수를 받아서 합을 계산한 후 HTML 문서로 만들어서 출력해야 한다.
- <FORM> 엘리먼트를 통해 입력된 데이터는 doGet, doPost 메서드의 첫 번째 파라미터 인 HttpServletRequest 타입의 파라미터에 대해 getParameter 메서드를 호출해서 가져올 수 있다.
 - 각 <INPUT> 서브엘리먼트를 통해 입력된 데이터를 가져오기 위해서는 다음과 같은 메서드를 호출해야 한다.
 - 이 메서드가 리턴하는 값은 수치 타입이 아니라 문자열 타입이다.

```
String str = request.getParameter( "NUM1 ");
```

<INPUT> 엘리먼트의 NAME 애트리뷰트 값

웹 브라우저로부터 데이터 입력받기

- 덧셈을 하기 위해서는 문자열 데이터를 수치 타입으로 변환해야 한다.
- 문자열을 int 타입으로 변환하기 위해서는 Integer 클래스의 parseInt 메서드를, double 타입으로 변환하기 위해서는 Double 클래스의 parseDouble 메서드를 이용하면 된다.

```
int num = Integer.parseInt(str);
```

String 타입의 데이터를
int 타입으로 변환하는 메서드

웹 브라우저로부터 데이터 입력받기

두 수의 합을 구하는 서블릿 클래스

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;

public class AdderServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        String nu1 = request.getParameter("num1");
        String nu2 = request.getParameter("num2");
        int num1 = Integer.parseInt(nu1);
        int num2 = Integer.parseInt(nu2);
        int sum = num1 + num2;
        PrintWriter out = response.getWriter();
        out.println("<html> <body>");
        out.printf("%d + %d = %d", num1, num2, sum);
        out.println("</body> </html>");    }
}
```

웹 브라우저로부터 데이터 입력받기

- 입력 데이터가 클 경우에는 URL 뒷부분의 데이터가 잘려나갈 수 있으므로 URL이 아닌 별도의 영역을 통해 입력 데이터를 전송해야 한다.
- <FORM> 엘리먼트의 시작 태그에 METHOD라는 애트리뷰트를 추가하고, 애트리뷰트 값으로 POST를 지정하면 된다.

입력 데이터가 URL이 아닌 별도의 영역을 통해
전송되도록 만드는 METHOD 애트리뷰트 값

<FORM ACTION =/brain/bbs-post METHOD=POST>

이름 : <INPUT TYPE=TEXT NAME=WRITER>

제목 : <INPUT TYPE=TEXT NAME=TITLE>

<TEXTAREA NAME=CONTENT> </TEXTAREA>

<INPUT TYPE=SUBMIT VALUE= '저장' >

<INPUT TYPE=RESET VALUE= '취소' >

</FORM>

게시판 글쓰기 기능의 데이터 입력을 위한 HTML 문서

```
<html>  <head>
<script type= "text/javascript">
  function chk() {
    if (!frm.title.value) { alert("제목");
      frm.title.focus(); return false; }
    if (!frm.name.value) { alert("이름");
      frm.name.focus(); return false; }
    if (!frm.content.value) { alert("내용");
      frm.content.focus(); return false; }
    return true;
  }
</script> </head> <body>
  <h1>게시판</h1>
  <form action= "Board" name= "frm" method= "post" onsubmit= "return chk()">
    <table border= "1" bgcolor= "pink">
      <tr> <td>제목</td> <td><input type= "text" name= "title"> </td> </tr>
      <tr> <td>이름</td> <td><input type= "text" name= "name"> </td> </tr>
      <tr> <td>내용</td> <td><textarea name= "content" rows= "10"
        cols= "30"> </textarea> </td> </tr>
      <tr> <td><input type= "submit" value= "확인"> </td>
        <td><input type= "reset" value= "취소"> </td> </tr>
    </table> </form>
</body>
</html>
```


■ 입력 데이터를 처리하는 서블릿 클래스의 작성 방법

- doGet 메서드를 선언하는 대신 doPost 메서드를 선언해야 한다. 웹 컨테이너는 POST라는 단어가 붙은 URL을 받으면 doGet 메서드가 아니라 doPost 메서드를 호출하기 때문이다
- doPost 메서드는 doGet 메서드와 마찬가지로 public 키워드를 붙여서 선언해야 하고, 파라미터 변수, 리턴 타입, 익셉션 타입도 doGet 메서드와 동일하다.

```
public class BBSPostServlet extends HttpServlet {  
    public void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
    }  
}
```

doGet 메서드와 리턴 타입, 파라미터 변수, 익셉션 타입이 동일합니다

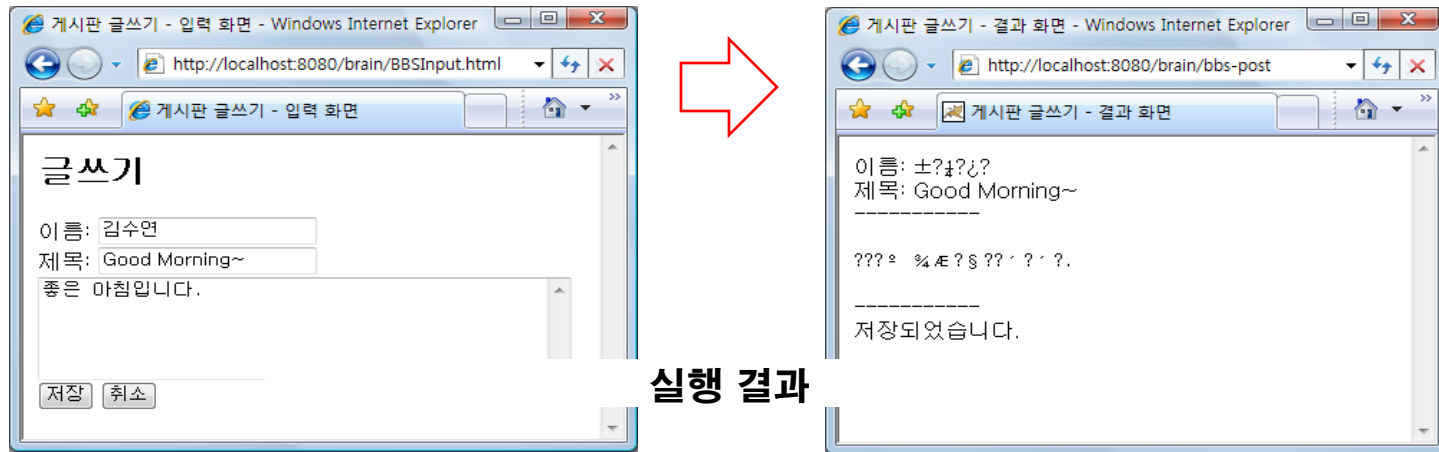
- doPost 메서드 안에서 입력 데이터를 가져오는 방법과 HTML 문서를 출력하는 방법도 doGet 메서드의 경우와 동일하다.

■ 두 번째 화면을 구현하는 서블릿 클래스

게시판 글쓰기 기능을 처리하는 서블릿 클래스 - 미완성

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class BBSPostServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        String title = request.getParameter("title");
        String name = request.getParameter("name");
        String content = request.getParameter("content");
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>게시판</h1>");
        out.println("제목 : " + title + "<p>");
        out.println("이름 : " + name + "<p>");
        out.println("내용<br><pre>" + content);
        out.println("</pre></body></html>");
        out.close();
    }
}
```

- 위 예제는 다음과 같이 한글 데이터의 입력 처리가 제대로 되지 않는다.



- 문제 해결: doPost 메서드 안에서 한글 데이터를 올바르게 가져오려면 첫 번째 파라미터 인 HttpServletRequest 파라미터에 대해 setCharacterEncoding 이라는 메서드를 호출해야 한다.

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

한글 코드 이름

- **setCharacterEncoding 메서드는 getParameter 메서드보다 반드시 먼저 호출해야 한다.**

게시판 글쓰기 기능을 처리하는 서블릿 클래스 - 완성

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class BBSPostServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        request.setCharacterEncoding("utf-8");
        String title = request.getParameter("title");
        String name = request.getParameter("name");
        String content = request.getParameter("content");
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>게시판</h1>");
        out.println("제목 : " + title + "<p>");
        out.println("이름 : " + name + "<p>");
        out.println("내용<br><pre>" + content);
        out.println("</pre></body></html>");
        out.close();
    }
}
```

TodayMenu.html

```
<HTML>
<HEAD>
<TITLE> SELECT & POST </TITLE>
</HEAD>
<BODY>
  <h3>오늘점심은 무엇을 먹을까?(2개이상 선택)</h3>
  <form method="post" action="TodayMenu">
    <select name="lunch" multiple size=2>
      <option>떡볶기
      <option>버섯덮밥
        <option>칼국수
        <option>치즈김밥
        <option>피자
    </select>
    <input type="submit">
  </form>
</BODY>
</HTML>
```

```
public class Ome extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        String[] menu = request.getParameterValues("menu");
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<html><body><h2>오늘 먹고 싶은 것</h2>");
        for(String m : menu) {
            out.println(m+"<br>");
        }
        out.println("</body></html>");
        out.close();
    }
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        doGet(request, response);
    }
}
```

- 이름, 아이디 항목은 <INPUT> 엘리먼트의 TYPE 애트리뷰트 값을 TEXT로 지정해서 만들 수 있다. 패스워드는 <INPUT> 엘리먼트의 TYPE 애트리뷰트 값을 PASSWORD로 지정해서 만든다.

```
<INPUT TYPE=TEXT NAME=NAME>
<INPUT TYPE=TEXT NAME=ID>
<INPUT TYPE=PASSWORD
NAME=PASSWORD>
```



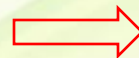
김현수

kimhs77

●●●●

- PASSWORD 타입으로 입력한 문자는 모니터상에 나타나지 않는다.
- 성별 항목은 라디오 버튼으로 만들어야 한다. 라디오 버튼은 <INPUT> 엘리먼트의 TYPE 애트리뷰트 값을 RADIO로 지정해서 만들 수 있으며, 반드시 NAME, VALUE 애트리뷰트를 써야 한다.

```
<INPUT TYPE=RADIO NAME=GENDER
VALUE=MALE>
<INPUT TYPE=RADIO NAME=GENDER
VALUE=FEMALE>
```



남 ☐ 여 ☒

↑
똑같은 NAME
애트리뷰트 값

↑
각각 다른 VALUE
애트리뷰트 값

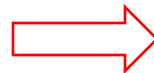
↑
한 항목을 선택하면 다른
항목의 선택이 해제된다.

❖ 다양한 형태로 데이터 입력받기

- 메일 수신 여부 항목은 체크 박스로 만들어야 한다. 체크 박스는 <INPUT> 엘리먼트의 애트리뷰트 값을 CHECKBOX로 지정해서 만들 수 있으며, NAME 애트리뷰트를 써야 한다. NAME 애트리뷰트에는 각각 다른 값을 지정해야 한다.

```
<INPUT TYPE=CHECKBOX NAME=INOTICE>  
<INPUT TYPE=CHECKBOX NAME=CNOTICE>  
<INPUT TYPE=CHECKBOX NAME=DNOTICE>
```

각각 다른 VALUE
애트리뷰트 값

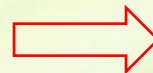


공지 메일 ☒ 광고 메일 ☐ 배송 확인 메일 ☒

한번 클릭하면 선택되고
또 한번 클릭하면 해제된다.

- 직업 항목은 선택 상자로 만들어야 한다. 선택 상자는 <SELECT> 엘리먼트를 이용해서 만들 수 있고, 이 엘리먼트의 시작 태그와 끝 태그 사이에 선택 항목의 이름을 포함한 <OPTION> 서브엘리먼트들을 써야 한다.

```
<SELECT NAME=JOB>  
  <OPTION>회사원</OPTION>  
  <OPTION>학생</OPTION>  
  <OPTION>기타</OPTION>  
</SELECT>
```



개인 정보를 입력받는 HTML 문서

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script type="text/javascript">
    function chk() {
        if (!frm.name.value) {alert("이름을 입력하세요");
            frm.name.focus();return false;}
        if (!frm.id.value) {alert("아이디를 입력하세요");
            frm.id.focus();return false;}
        if (!frm.pass.value) {alert("암호를 입력하세요");
            frm.pass.focus();return false;}
        if (!frm.pass2.value) {alert("암호확인을 입력하세요");
            frm.pass2.focus();return false;}
        if (frm.pass.value != frm.pass2.value) { alert("암호가 틀려 !");
            frm.pass.focus(); frm.pass.value = "";
            frm.pass2.value = ""; return false;}
        var chk = false;
        for (i =0 ; i < frm.gender.length; i++) {
            if (frm.gender[i].checked == true) {    chk = true; break;    }
        }
        if (chk==false) { alert("성별 체크 안했어"); return false; }
        return true;
    }
</script></head>
```

```
<body>
<form action="Person" name="frm" method="post" onsubmit="return chk()">
<table border="1" bgcolor="pink" align="center">
  <caption><h1>개인 정보</h1></caption>
  <tr><td>이름</td><td><input type="text" name="name"></td></tr>
  <tr><td>아이디</td><td><input type="text" name="id"></td></tr>
  <tr><td>암호</td><td><input type="password" name="pass"></td></tr>
  <tr><td>암호확인</td><td><input type="password" name="pass2"></td></tr>
  <tr><td>성별</td>
    <td><input type="radio" NAME="gender" VALUE="남자">
      남
      <input type="radio" NAME="gender" VALUE="여자">
      여
    </td></tr>
  <tr><td>메일 수신</td>
    <td><input type="checkbox" NAME="inotice" VALUE="공지">
      공지
      <input type="checkbox" NAME="cnotice" VALUE="광고">
      광고
      <input type="checkbox" NAME="dnotice" VALUE="배송">
      배송
    </td></tr>
  <tr><td>직업</td><td><select NAME="job">
    <option value="회사원">회사원</option><option value="수강생">수강생</option>
    <option value="기타">기타</option></select></td></tr>
  <tr><td><input type="submit" value="확인"></td>
    <td><input type="reset" value="취소"></td></tr>
</table></form>
</body>
</html>
```

■ 서블릿 클래스의 작성 방법

- <INPUT> 엘리먼트의 TYPE 애트리뷰트 값이 TEXT 또는 PASSWORD일 경우 다음과 같은 방법으로 입력 데이터를 가져올 수 있다.

```
<INPUT TYPE=TEXT NAME=NAME>  
<INPUT TYPE=TEXT NAME=ID>  
<INPUT TYPE=PASSWORD NAME=PASSWORD>
```



```
String name = request.getParameter( "NAME ");  
String id = request.getParameter( "ID ");  
String password = request.getParameter( "PASSWORD ");
```

텍스트 상자에 입력된 값

NAME 애트리뷰트 값

- 라디오 버튼의 경우 동일한 NAME 애트리뷰트 값을 갖는 모든 라디오 버튼에 대해 `getParameter` 메서드를 한 번만 호출해야 한다.

```
<INPUT TYPE=RADIO NAME=GENDER VALUE=MALE>  
<INPUT TYPE=RADIO NAME=GENDER VALUE=FEMALE>
```



```
String gender = request.getParameter( "GENDER " );
```

선택된 항목의
VALUE 애트리뷰트 값

NAME 애트리뷰트
값

- 체크 박스의 경우 각각의 체크 박스에 대해 `getParameter` 메서드를 한 번씩 호출해야 한다.

```
<INPUT TYPE=CHECKBOX NAME=INOTICE>  
<INPUT TYPE=CHECKBOX NAME=CNOTICE>  
<INPUT TYPE=CHECKBOX NAME=DNOTICE>
```



```
String iNotice = request.getParameter( "INOTICE " );  
String cNotice = request.getParameter( "CNOTICE " );  
String dNotice = request.getParameter( "DNOTICE " );
```

'on' 또는 null

NAME 애트리뷰트 값

개인 정보를 입력받는 HTML 문서

```
import javax.servlet.http.*; import javax.servlet.*; import java.io.*;
public class PersonalInfoServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        String name = request.getParameter("name");
        String id = request.getParameter("id");
        String pass = request.getParameter("pass");
        String gender = request.getParameter("gender");
        String inotice = request.getParameter("inotice");
        String cnotice = request.getParameter("cnotice");
        String dnotice = request.getParameter("dnotice");
        String job = request.getParameter("job");
        if (inotice==null) inotice = "";      if (cnotice==null) cnotice = "";
        if (dnotice==null) dnotice = "";
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<html><body><h1>개인 정보 </h1>");
        out.println("이름 : " + name + "<p>아이디 : " + id + "<p>");
        out.println("암호 : " + pass + "<p>성별 : " + gender + "<p>");
        out.println("메일수신 : "+inotice+" "+cnotice+" "+dnotice+"<p>");
        out.println("직업 : " + job + "<p>");
        out.println("</body></html>");
        out.close();
    }
}
```

톰캣기반에서의 한글처리

1. 서버에서 웹 브라우저에 응답되는 페이지의 화면 출력 시 한글처리
 - `<%@ page contentType="text/html;charset=utf-8"%>`
2. 웹 브라우저에서 서버로 넘어오는 파라미터 값에 한글이 있는 경우 (Post방식) 한글처리
 - `<% request.setCharacterEncoding("utf-8");%>`
3. 웹 브라우저에서 서버로 넘어오는 파라미터 값에 한글이 있는 경우 (Get방식) 한글처리
 - 톰캣홈\conf 폴더와 이클립스의 [프로젝트 탐색기]뷰에서 [Servers]-[Tomcat v5.5 서버]항목 에 있는 server.xml 파일 둘다 <Connector>태그의 속성에 `URIEncoding="EUC-KR"`문장을 추가.

```
<Connector port="8181" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" redirectPort="8443" acceptCount="100"
    connectionTimeout="20000" disableUploadTimeout="true"
    URIEncoding="utf-8" />
```

- `new String(ko.getBytes("utf-8"), "8859_1");`
- `new String (en.getBytes("8859_1"), "utf-8");`
- ※. Server.xml Connector태그에 `useBodyEncodingForURI= "true"` 추가
서블릿에서 `request.setCharacterEncoding("utf-8");`

```
import java.io.*;
public class HangulConversion {
    public static String toEng (String ko) {
        if (ko == null) { return null; }
        try {
            return new String(ko.getBytes("euc-kr"), "8859_1");
        } catch (Exception e) {
            return ko;
        }
    }
    public static String toKor (String en) {
        if (en == null) { return null; }
        try {
            return new String (en.getBytes("8859_1"), "euc-kr");
        } catch (Exception e) {
            return en;
        }
    }
}
```