



2 서블릿의 기초

뇌를 자극하는 JSP & Servlet

❖ 학습목표

- 서블릿 클래스는 자바 클래스 형태로 구현되는 웹 애플리케이션 프로그램이며, 일반적인 자바 클래스를 작성 할 때보다 지켜야 할 규칙이 많다. 이 장에서는 그 규칙들을 배워보자.

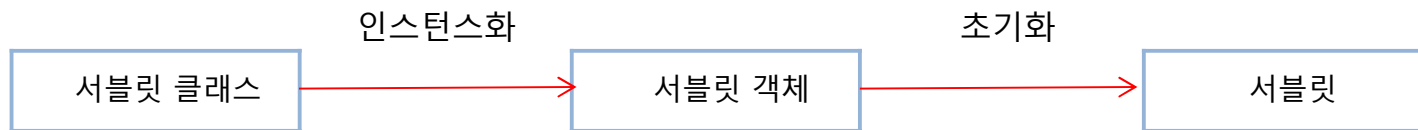
❖ 내용

- 서블릿이란?
- 서블릿 클래스의 작성, 컴파일, 설치, 등록
- 톰캣 관리자 프로그램 사용하기
- 웹 브라우저로부터 데이터 입력받기



1. 서블릿이란?

- 서블릿이란 서블릿 클래스로부터 만들어진 객체이다.
- 웹 컨테이너는 서블릿 클래스를 가지고 서블릿 객체를 만든 다음 그 객체를 초기화해서 웹 서비스를 할 수 있는 상태로 만드는데, 이 작업을 거친 서블릿 객체만 서블릿이라고 할 수 있다.



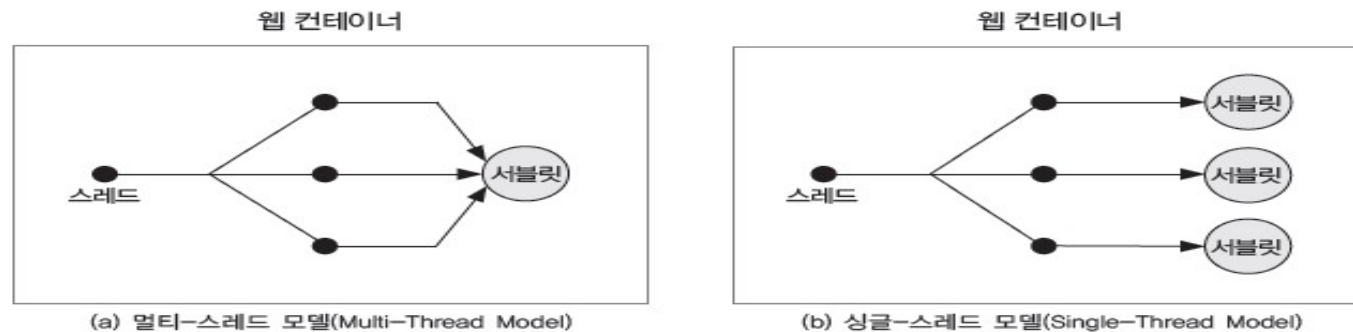
[그림 2-1] 서블릿 클래스, 서블릿 객체, 서블릿

- 인스턴스화(**instantiation**)란 클래스를 가지고 객체를 만드는 행위를 말한다.
- 멀티스레드(**multithread**)란 프로그램의 실행 흐름이 여러 갈래(**thread**)로 나뉘어져서 동시에 실행되는 것을 말한다.



1. 서블릿이란?

- 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.



[그림 2-2] 멀티-스레드 모델과 싱글-스레드 모델

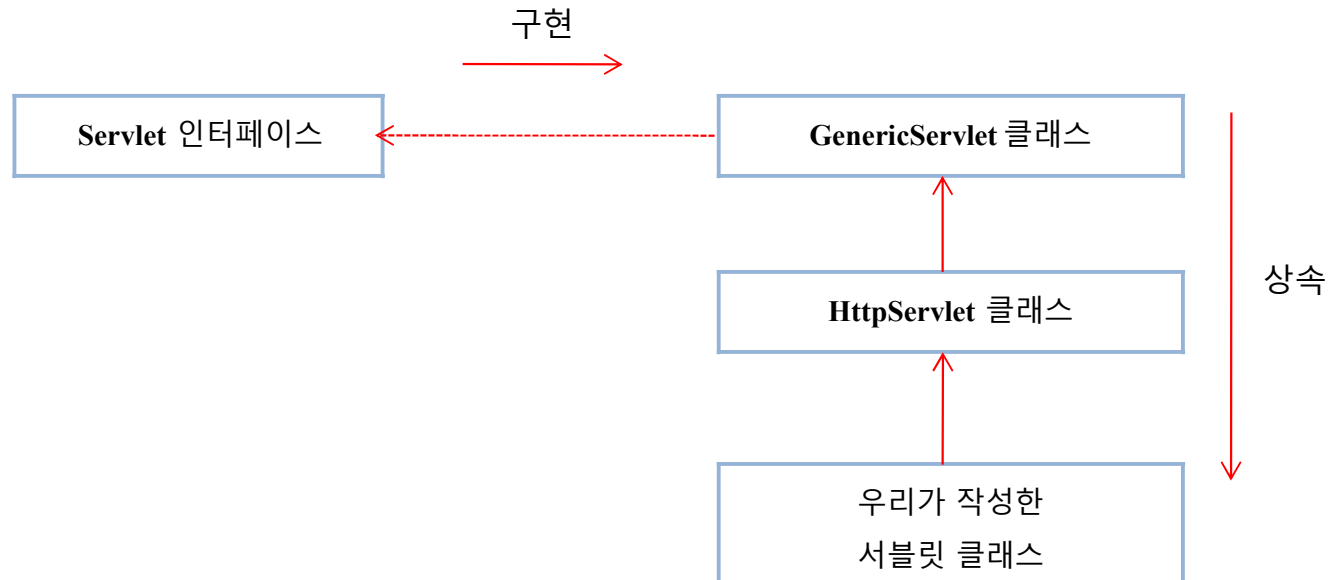
- 싱글-스레드 모델에서는 데이터 공유 문제를 걱정할 필요가 없지만 시스템 자원과 메모리가 더 많이 소모된다.



2. 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

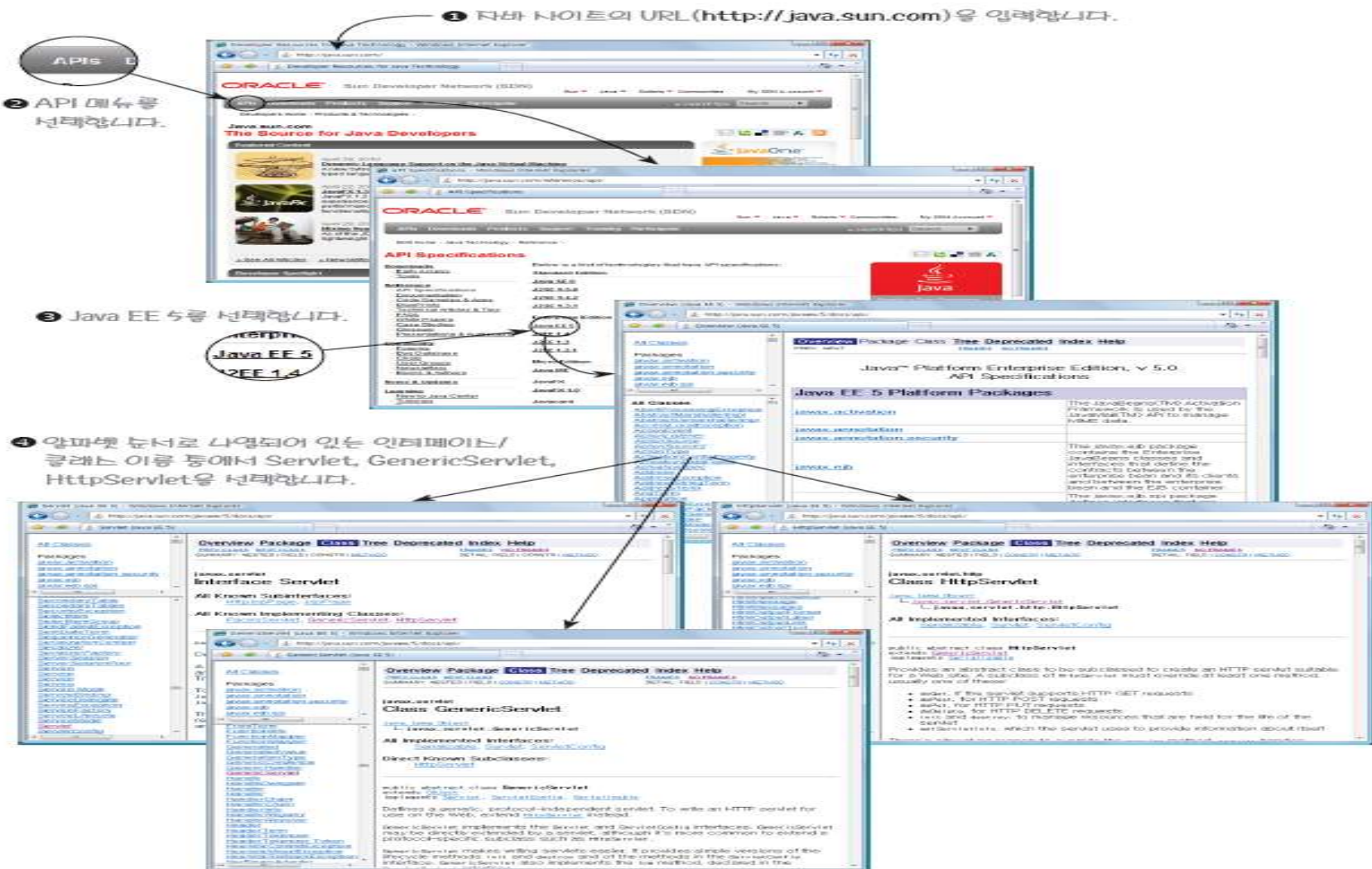


[그림 2-3] 서블릿 클래스의 상속/구현 관계



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스의 작성을 위한 준비



[그림 2-4] Servlet 인터페이스와 GenericServlet, HttpServlet 클래스의 API 규격서

2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 작성하기

- 서블릿 클래스를 작성할 때 지켜야 할 첫 번째 규칙: **javax.servlet.http.HttpServlet** 클래스를 상속받도록 만들어야 한다. 그리고 **public** 클래스로 만들어야 한다.

```
public class HundredServlet extends HttpServlet {  
  
}
```

프로그래머가
정한 클래스 이
름

서블릿 클래스
의
슈퍼클래스

- 서블릿 클래스 안에 **doGet** 또는 **doPost** 메서드를 선언해야 하며, 이 두 메서드는 **javax.servlet.http.HttpServletRequest**와 **javax.servlet.http.HttpServletResponse** 타입의 파라미터를 받고, 메서드 밖으로 **javax.servlet.ServletException**과 **java.io.IOException**을 던질 수 있도록 선언해야 한다.



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 작성하기

HttpServlet 클래스의 API 규격서

1 HttpServlet 클래스의 API 규격서에서 링크를 바를 내리세요.

2 doGet 메서드의 이름을 클릭하면 메서드의 상세 정보가 나옵니다.

3 doGet 메서드의 리턴 타입, 파라미터 변수, 익셉션 타입을 확인하세요.

```
void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
```

Called by the server (via the service method) to allow a servlet to handle a GET request.

Overriding this method to support a GET request also automatically supports an HTTP HEAD request. A HEAD request is a GET request that returns no body in the response; only the request header fields.

When overriding this method, read the request data, write the response headers, get the response's writer or output stream object, and finally, write the response data. It's best to include content type and encoding. When using a PrintWriter object to return the response, set the content type before accessing the PrintWriter object.

The servlet container must write the headers before committing the response, because in HTTP the headers must be sent before the response body.

Where possible, set the Content-Length header (with the `ServletResponse.setContentLength(int)` method), to allow the servlet container to use a persistent connection to return its response to the client, improving performance. The content length is automatically set if the entire response fits inside the response buffer.

When using HTTP 1.1 chunked encoding (which means that the response has a Transfer-Encoding header), do not set the Content-Length header.

The GET method should be safe, that is, without any side effects for which users are held responsible. For example, most form queries have no side effects. If a client request is intended to change stored data, the request

[그림 2-5] HttpServlet 클래스의 doGet 메서드

2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 작성하기

- **doGet** 메서드를 작성할 때는 다음과 같은 골격을 만드는 것으로 시작한다.

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

↑
HttpServlet 클래스의 doGet 메서드와
리턴 타입, 파라미터 변수, 익셉션 타입이 동일해야
한다.

- **doGet** 메서드를 **public**으로 선언해야 하는 이유는 웹 컨테이너가 웹 브라우저로부터 요청을 받아서 메서드를 호출할 때 필요하기 때문이다.
- **doGet** 메서드의 **throws** 절에 있는 **ServletException**과 **IOException**이 필요치 않으면 생략할 수도 있다. 하지만 다른 익셉션을 추가할 수는 없다.



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 작성하기

- **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 **getWriter**라는 메서드를 호출해서 **PrintWriter** 객체를 구한다.

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

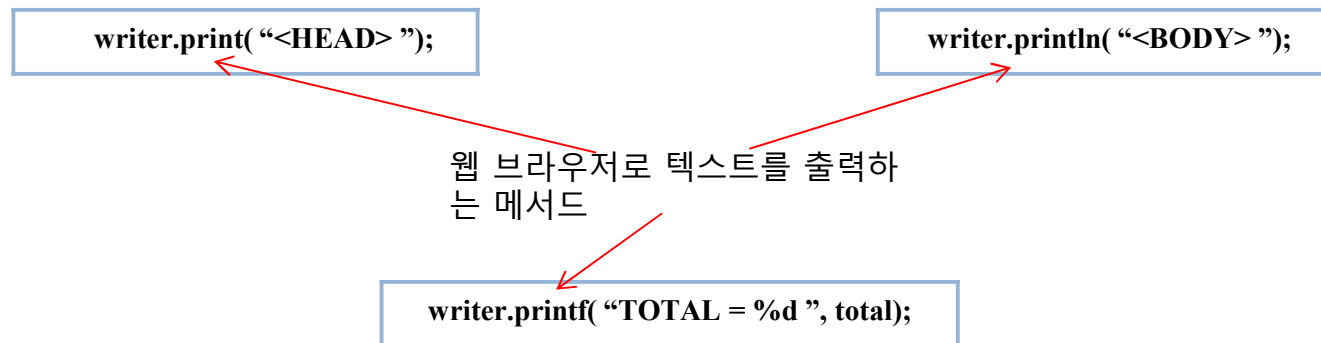
PrintWriter 객체를 리턴하는 메
서드



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 작성하기

- **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.



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 작성하기

- 계산 결과를 웹 브라우저로 출력하는 코드

```
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> ");  
    }  
}
```

계산 결과를 HTML로 만들어서 웹 브라우저로 출력하는 명령문



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 작성하기

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

[예제2-1] 1부터 100까지 더하는 서블릿 클래스

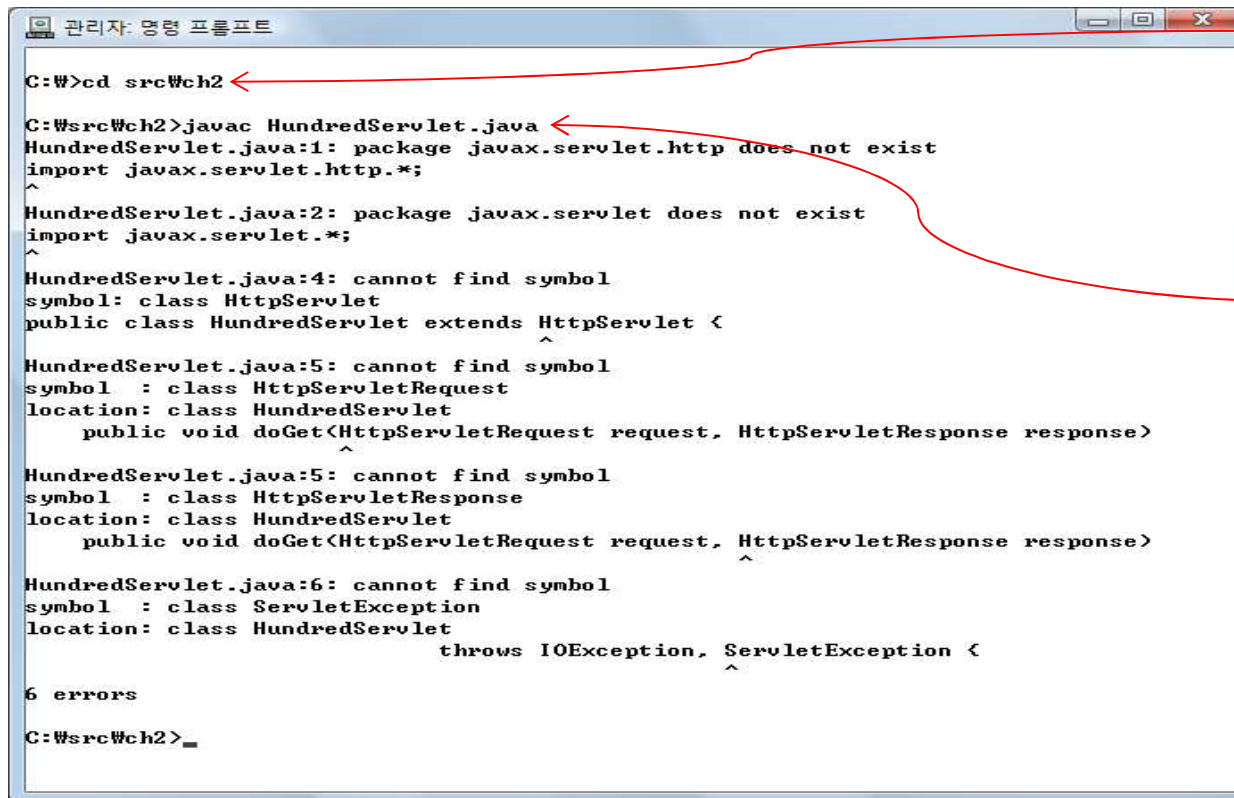
```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class HundredServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int total = 0;
        for (int cnt = 1; 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> ");
    }
}
```



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 컴파일하기

- 서블릿 클래스도 자바 클래스와 마찬가지로 **javac** 명령을 이용해서 컴파일할 수 있다.
- [예제 2-1]의 소스 코드를 저장해 놓는 디렉터리로 가서 **javac** 명령으로 컴파일하면 처음에는 다음과 같은 에러 메시지가 나온다.



```
C:\srcWch2>cd srcWch2
C:\srcWch2>javac HundredServlet.java
HundredServlet.java:1: package javax.servlet.http does not exist
import javax.servlet.http.*;
^
HundredServlet.java:2: package javax.servlet does not exist
import javax.servlet.*;
^
HundredServlet.java:4: cannot find symbol
symbol: class HttpServlet
public class HundredServlet extends HttpServlet {
^
HundredServlet.java:5: cannot find symbol
symbol : class HttpServletRequest
location: class HundredServlet
    public void doGet(HttpServletRequest request, HttpServletResponse response)
^
HundredServlet.java:5: cannot find symbol
symbol : class HttpServletResponse
location: class HundredServlet
    public void doGet(HttpServletRequest request, HttpServletResponse response)
^
HundredServlet.java:6: cannot find symbol
symbol : class ServletException
location: class HundredServlet
        throws IOException, ServletException {
^
6 errors
C:\srcWch2>
```

소스 코드가 있는 디렉터리로 이동하는 명령

서블릿 클래스를 컴파일하는 명령

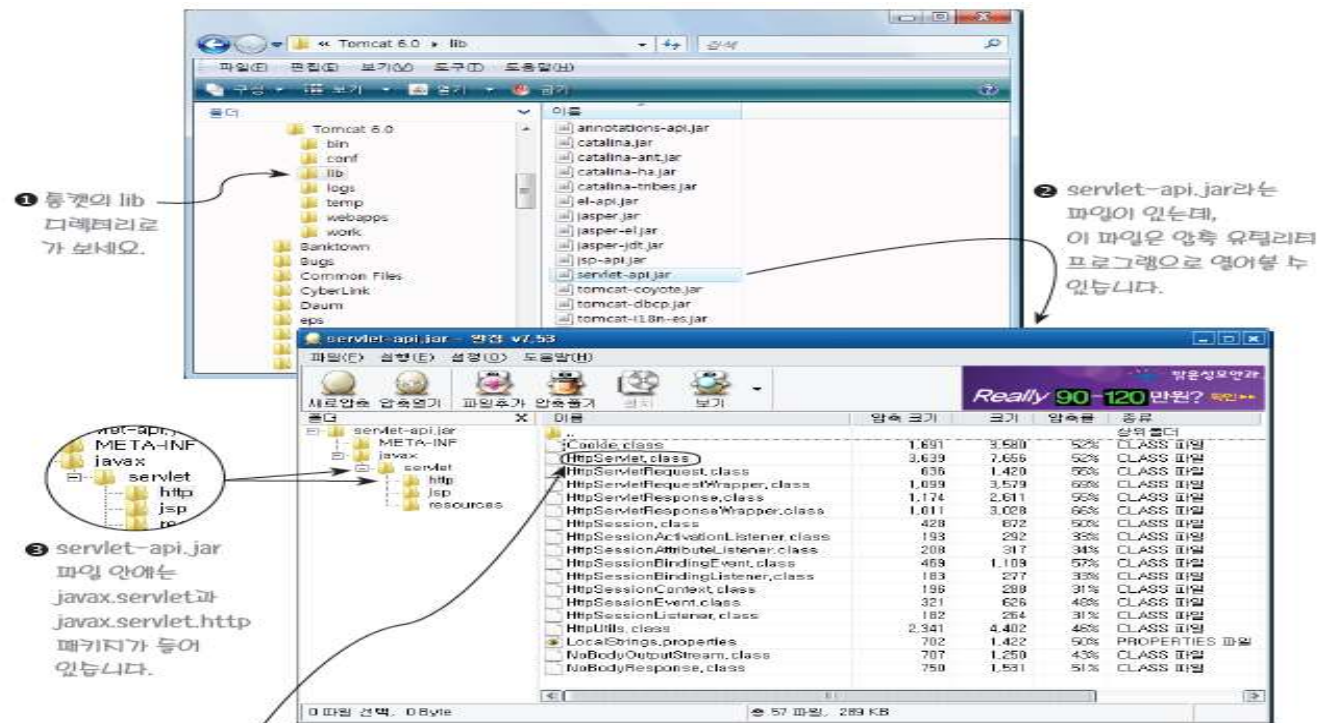
[그림 2-6] 아무 옵션도 사용하지 않고 서블릿 클래스를 컴파일했을 때 나오는 에러 메시지



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 컴파일하기

- 에러 메시지가 나오는 이유는 **import**한 **javax.servlet**과 **javax.servlet.http** 패키지가 **JDK**의 표준 라이브러리 안에 없기 때문이다.
- 서블릿 클래스를 컴파일할 때는 **-cp** 옵션을 이용해서 이 두 패키지가 속하는 라이브러리의 경로명을 명시해 주어야 한다.



javax.servlet.http 패키지에 속하는 HttpServlet 클래스의 파일입니다

[그림 2-7] javax.servlet과 javax.servlet.http 패키지가 들어 있는 servlet-api.jar 파일



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 컴파일하기

- 서블릿 클래스를 컴파일할 때 **-cp** 옵션으로 **servlet-api.jar** 파일 경로명을 지정하면 컴파일 에러가 발생하지 않는다.



```
관리자: 명령 프롬프트
C:\src\ch2>javac -cp "C:\Program Files\Apache Software Foundation\Tomcat 6.0\lib\servlet-api.jar" HundredServlet.java
C:\src\ch2>
```

[그림 2-8] 서블릿을 컴파일하는 방법(1)

- 컴파일에 실패한다면 경로명을 입력 과정에서 실수일 수 있다.
- 톰캣의 **lib** 서브디렉터리에 있는 **servlet-api.jar** 파일을 **JDK** 설치 디렉터리 아래의 **jre\lib\ext** 서브디렉터리로 복사한다.



```
관리자: 명령 프롬프트
C:\src\ch2>javac HundredServlet.java
C:\src\ch2>
```

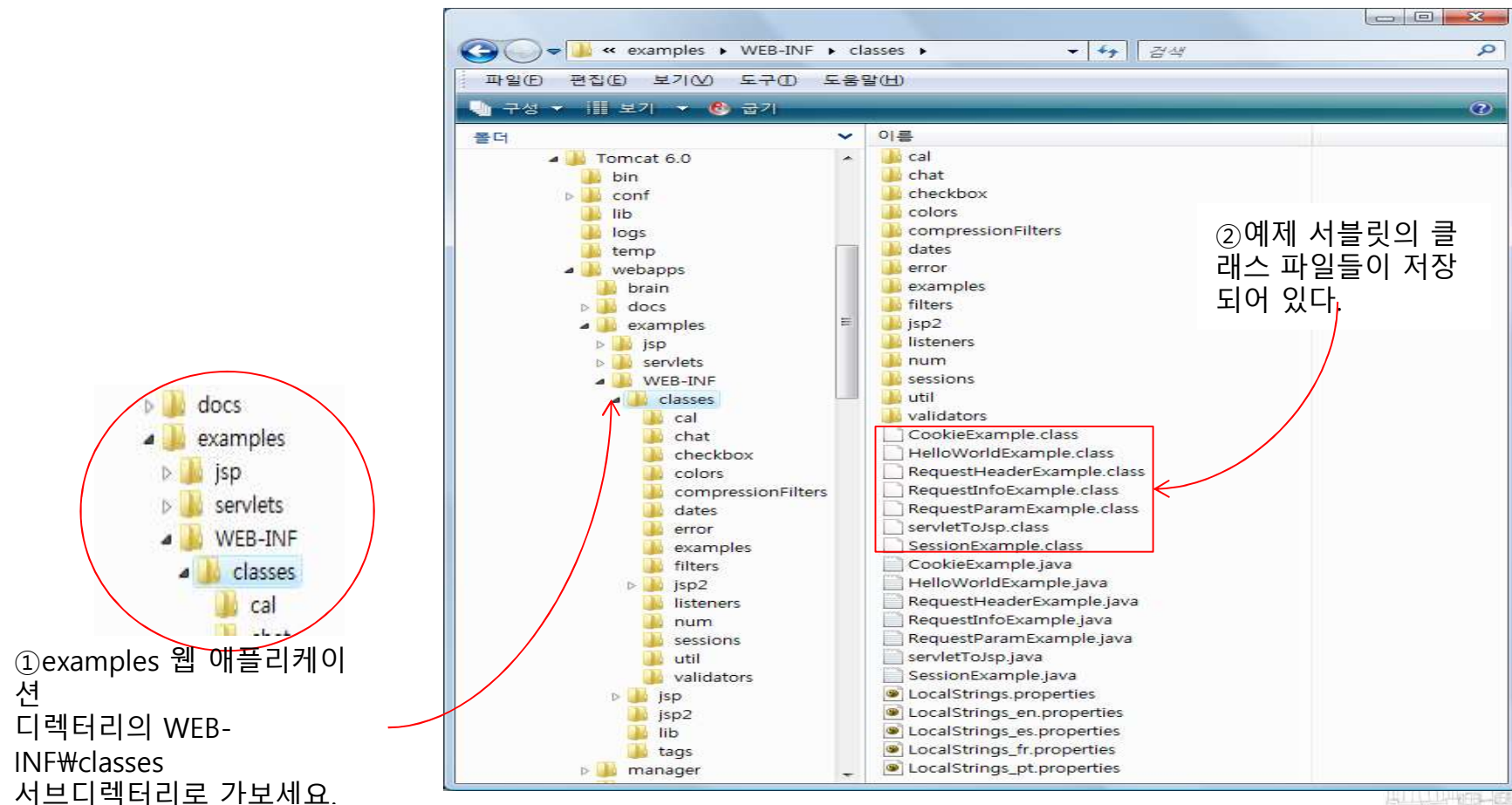
[그림 2-10] 서블릿을 컴파일하는 방법(2)



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 설치하기

- JSP 페이지와 달리 서블릿 클래스는 소스 코드를 설치할 필요가 없고, 컴파일 결과물인 클래스 파일만 설치하면 된다.

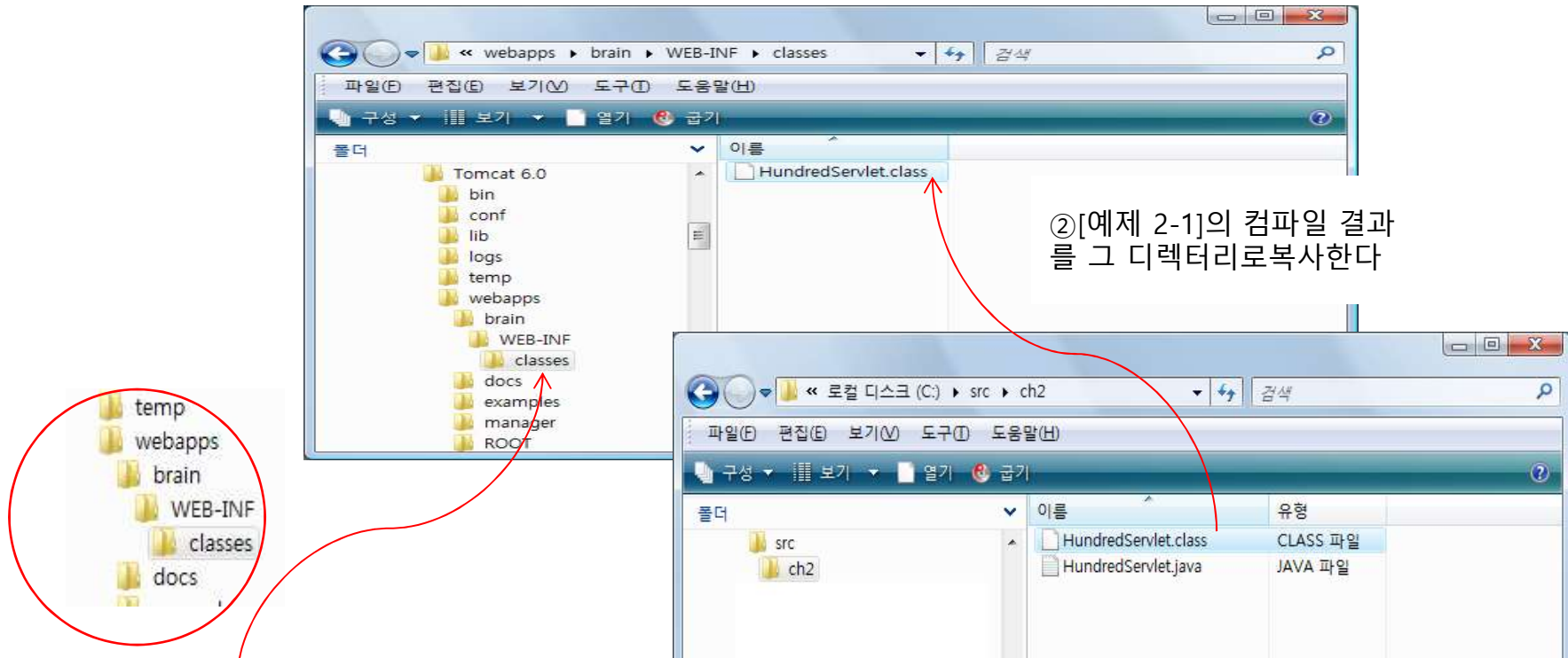


[그림 2-11] 예제 서블릿의 클래스 파일들이 저장되어 있는 디렉터리

2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 설치하기

- 톰캣의 **webapps\brain** 디렉터리로 가서 **WEB-INF**라는 서브디렉터리를 만들고, 그 아래에 **classes**라는 서브디렉터리를 만든 후, 컴파일 결과물인 **HundredServlet.class** 파일을 저장한다.



- ① brain 웹 애플리케이션 디렉터리
아래에 WEB-INF\classes라는
서브디렉터리를 만든다.



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 등록하기

- 서블릿 클래스는 **JSP** 페이지와 달리 설치뿐만 아니라 등록 과정도 필요하다.
- 웹 애플리케이션의 디플로이먼트 디스크립터 파일에 등록해야 한다.
- 웹 애플리케이션의 디플로이먼트 디스크립터 파일란 웹 애플리케이션 디렉터리의 **WEB_INF** 서브디렉터리 아래 있는 **web.xml**이라는 이름의 파일을 말한다.
예: 톰캣의 **webapps\examples\WEB_INF** 디렉터리에 있는 **web.xml** 파일
- **XML** 파일이고, 텍스트 에디터를 이용해서 열어 볼 수 있다.
- 주의: 톰캣에 있는 **web.xml** 파일 중에는 **UNIX** 포맷으로 만들어진 것도 있는데, 이런 파일은 메모장으로 열면 줄 바꿈 표시가 제대로 되지 않을 수 있다. **그럴 때는** 메모장 대신 워드패드처럼 기능이 풍부한 텍스트 에디터를 이용해서 여는 것이 좋다.



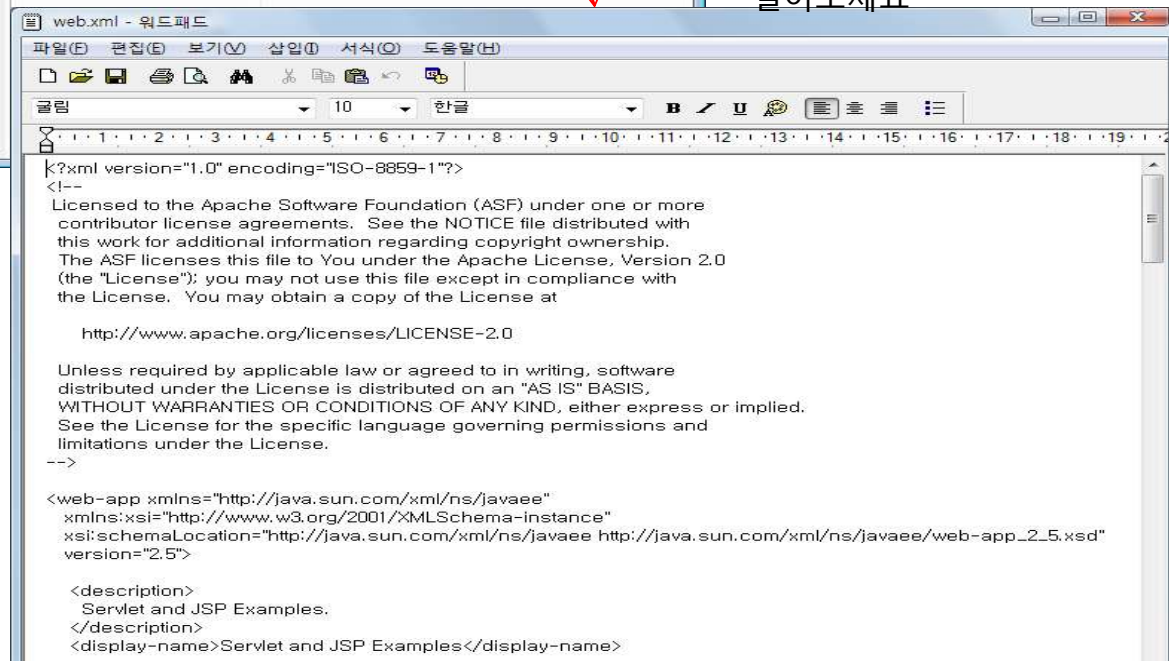
2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 등록하기



② 거기에 있는
web.xml
파일을 텍스트 에디터
로
열어보세요

① 톰캣의 examples 웹
애플리케이션의 WEB-
INF
서브디렉터리로 가세요.



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 알아두세요(XML 문법의 기초)

- XML은 HTML과 마찬가지로 텍스트 내용에 태그(또는 마크업)를 첨가하기 위해 사용되는 문법이다.
- XML과 HTML의 서로 다른 점은 다음과 같다.
 - 첫째 : XML 문서의 제일 앞에는 XML 선언이 올 수 있다. XML 선언은 XML 문서의 작성에 사용된 XML 규격서의 버전, XML 문서를 저장하는 데 사용된 문자 코드의 인코딩 방식을 표시하는 역할을 한다.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

XML 규격서의 버전

문자셋의 인코딩 방식

단, XML 문서가 ASCII 문자로만 구성되었을 경우에는 XML 선언을 생략할 수도 있다.

- 둘째 : HTML에서는 모든 HTML 문서의 작성 방법이 동일하지만, XML에서는 XML 문서의 종류에 따라 작성 방법이 달라질 수 있다.

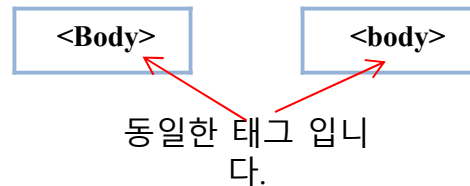


2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

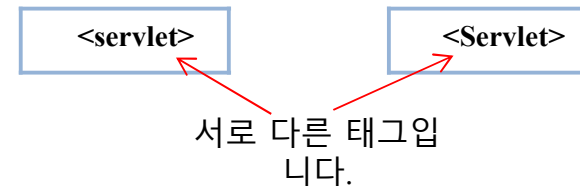
❖ 알아두세요(XML 문법의 기초)

- 셋째 : **HTML**에서는 엘리먼트 이름과 애트리뷰트 이름에 있는 대소문자를 구분하지 않지만 **XML**에서는 엄격하게 구분한다.

★HTML 문서★

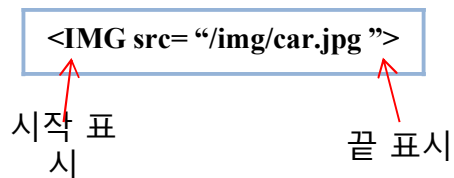


★XML 문서★

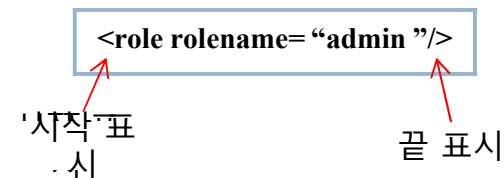


- 넷째 : **HTML**에서는 단독으로 사용되는 태그가 `<`로 시작해서 `>`로 끝나야 하지만, **XML**에서는 `<`로 시작해서 `/>`로 끝나야 한다.

★HTML 문서★



★XML 문서★



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 알아두세요(XML 문법의 기초)

- 다섯째 : **HTML**에서는 애트리뷰트 값을 따옴표로 묶지 않고 쓸 수도 있지만, **XML**에서는 반드시 따옴표로 묶어서 써야한다.

★HTML 문서★

```
<IMG src=/img/car.jpg>
```

올바른
문법

★XML 문서★

```
<role rolename=admin />
```

잘못된
문법

- 여섯째 : **HTML**에서는 다소 문법이 맞지 않는 부분이 있으면 웹 브라우저가 이를 보정해서 처리하지만, **XML**에서는 문법에 조금이라도 맞지 않는 부분이 있으면 **XML** 문서 전체가 올바르게 처리 되지 않는다.

★HTML 문서★

```
<H1>회사 개요</H2>
```

짜이 맞지 않아도
웹 브라우저가 보정해
준다

★XML 문서★

```
<servlet-name>MyServlet</servlet-class>
```

짜이 맞지 않으면 문서 전
체가 올바르게 처리되지
않는다.



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 등록하기

- **web.xml** 파일은 웹 애플리케이션 디렉터리마다 하나씩만 만들 수 있다.
- **web.xml** 파일을 새로 만들 때는 루트 엘리먼트인 **<web-app>**을 만드는 일부부터 시작하는 것이 좋다.

```
<web-app>  
</web-app>
```

web.xml 파일의 루트 엘리먼트

- 웹 서버가 웹 브라우저로부터 **URL**을 받았을 때 서블릿 클래스를 찾아서 호출하기 위해 필요한 두 정보는 다음의 두 엘리먼트 안에 기록해야 한다.

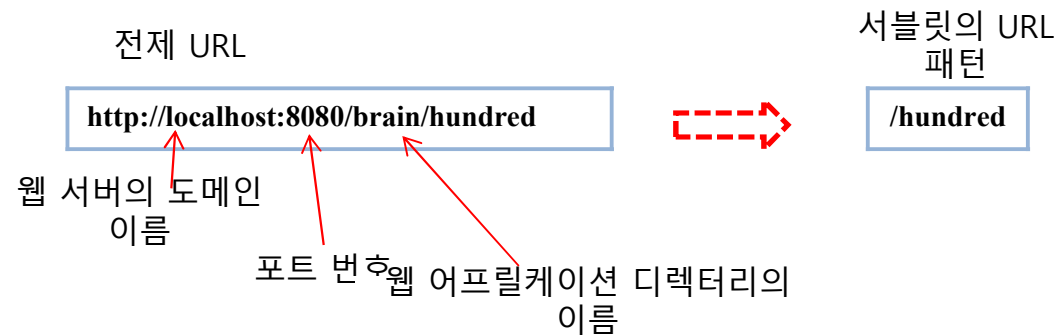
```
<web-app>  
  <servlet>  
    ← 서블릿 클래스의 이름이 들어갈 부분  
  </servlet>  
  <servlet-mapping>  
    ← 서블릿 클래스를 호출할 때 사용할 URL이 들어갈 부분  
  </servlet-mapping>  
</web-app>
```



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 등록하기

- **<servlet-mapping>** 엘리먼트 안에는 전체 **URL**이 아니라, 웹 서버의 도메인 이름, 포트 번호, 웹 어플리케이션 디렉터리 이름을 제외한 나머지 부분만 써야 한다.



```
<web-app>
  <servlet>
    <servlet-class>HundredServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <url-pattern>/hundred</url-pattern>
  </servlet-mapping>
</web-app>
```

Red arrows point from the XML code to labels: one from `<servlet-class>HundredServlet</servlet-class>` to '서블릿 클래스의 이름' (Servlet class name), and another from `<url-pattern>/hundred</url-pattern>` to '서블릿의 URL 패턴' (Servlet URL pattern).



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 등록하기

- **web.xml** 파일에는 나중에 다른 **<servlet>**, **<servlet-mapping>** 엘리먼트가 추가 될 수도 있으므로, 연관된 두 엘리먼트가 서로 찾을 수 있도록 연결시켜야 한다.
방법: **<servlet>** 엘리먼트 안에 서블릿 식별자를 쓰고, **<servlet-mapping>** 엘리먼트에서 그 식별자를 이용해서 **<servlet>** 엘리먼트를 참조하도록 만든다.

```
<web-app>
  <servlet>
    <servlet-name>hundred-servlet</servlet-name>
    <servlet-class>HundredServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>hundred-servlet</servlet-name>
    <url-pattern>/hundred</url-pattern>
  </servlet-mapping>
</web-app>
```

참조



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 등록하기

- 모든 **web.xml** 파일 안에 반드시 써 넣어야 하는 두 가지 정보

web.xml 파일의 작성에 사용된 문법의
식별자

```
<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_2_5.xsd"
  version="2.5">
```

그 문법의 버
전



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 클래스 등록하기

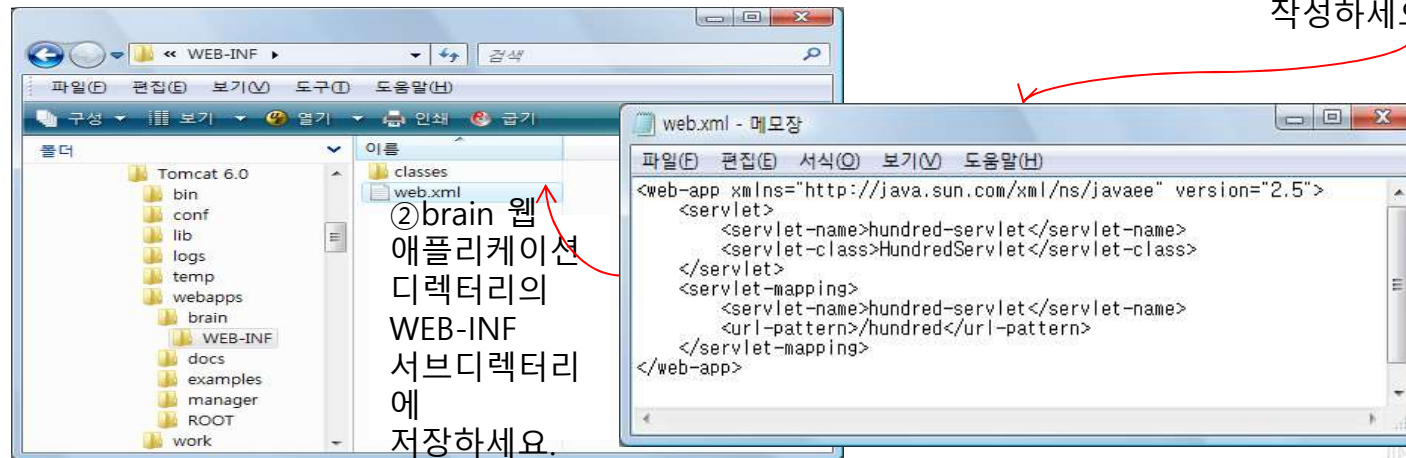
- **<web-app>** 엘리먼트의 시작 태그에 다음과 같은 내용을 추가하라.

web.xml 파일의 작성에 사용된 문법의
식별자

그 문법의
버전

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
  <servlet>
    <servlet-name>hundred-servlet</servlet-name>
    <servlet-class>HundredServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>hundred-servlet</servlet-name>
    <url-pattern>/hundred</url-pattern>
  </servlet-mapping>
</web-app>
```

① 텍스트 에디터로
web.xml 파일을
작성하세요

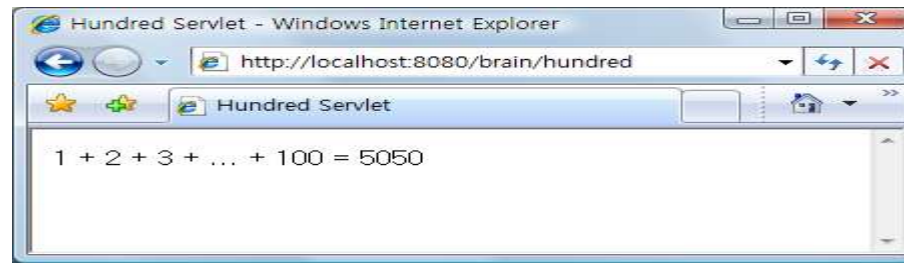


[그림 2-14] 서블릿 클래스를 web.xml 파일에 등록하는 방법

2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

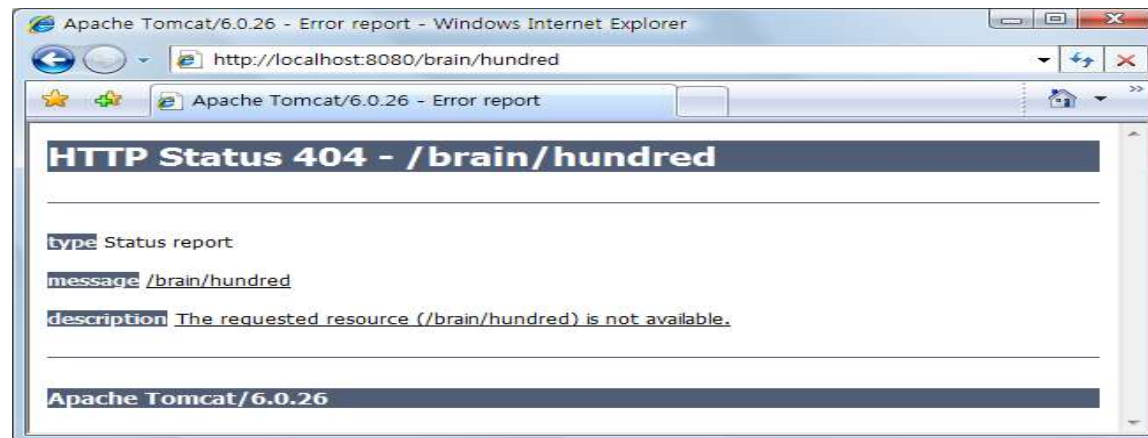
❖ 서블릿 실행하기

- 웹 브라우저를 열고 주소 창에 **http://localhost:8080/brain/hundred**라는 URL을 입력했을 때 나오는 [예제 2-1]의 정상적인 실행 결과이다.



[그림 2-15] 예제 2-1의 실행 결과 – 성공적인 결과

- 등록 과정이 잘못되었다면 다음과 같은 에러 페이지가 나타난다.



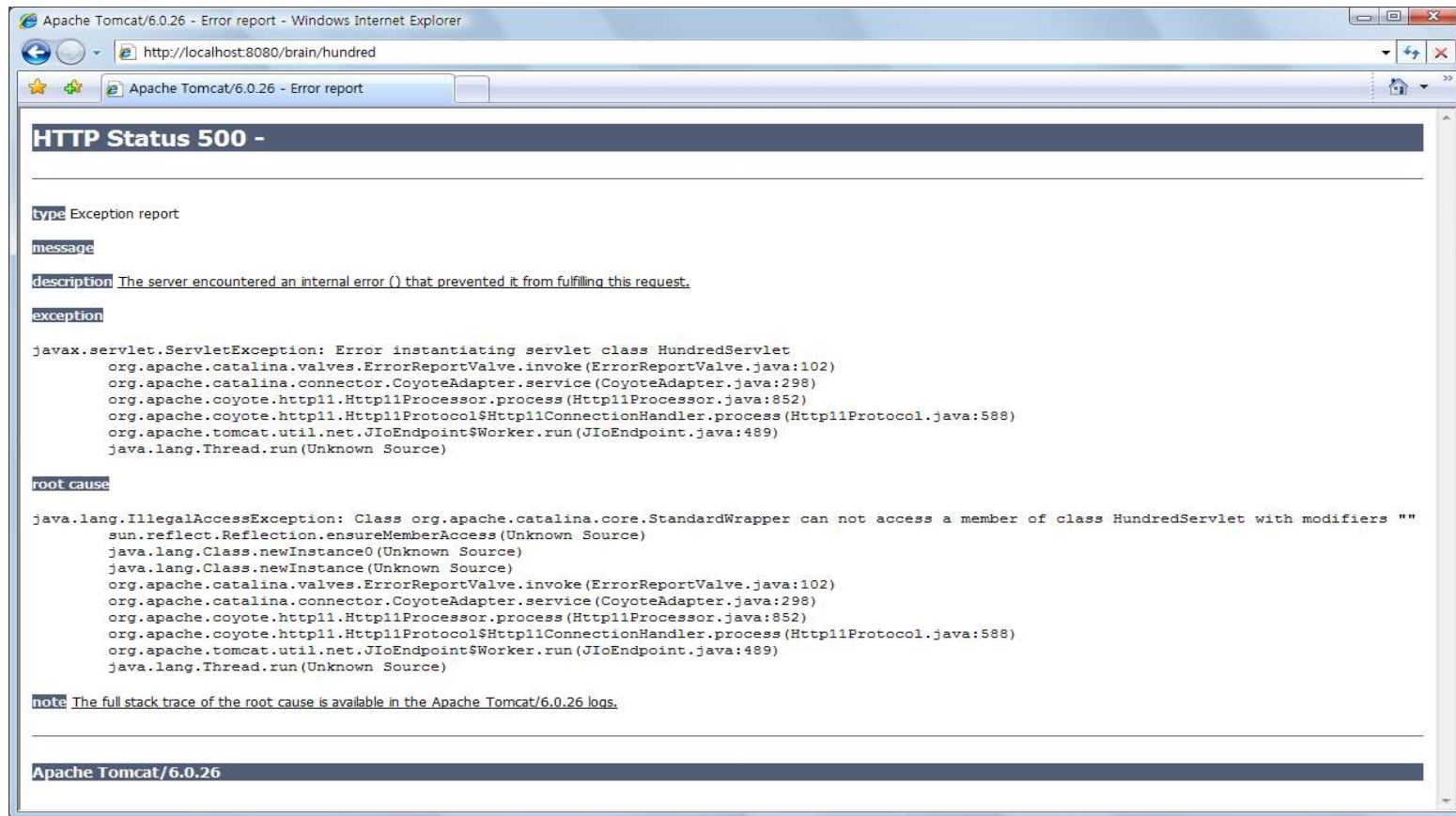
[그림 2-16] 예제 2-1의 실행 결과 – 서블릿의 등록이 잘못되었을 때



2. 서블릿 클래스의 작성, 컴파일, 설치, 등록

❖ 서블릿 실행하기

- 서블릿 클래스의 소스 코드에서 잘못된 부분이 있다면 다음과 같은 에러 페이지가 나타난다.



[그림 2-17] 예제 2-1의 실행 결과 - 서블릿 클래스의 소스 코드가 잘못되었을 때

3. 톰캣 관리자 프로그램 사용하기

❖ 톰캣 관리자 프로그램의 사용 방법

1 톰캣의 URL을 입력하세요.

2 'Tomcat Manager' 링크를 선택하며 로그인 창이 나타납니다.

localhost에 연결

Tomcat Manager Application의 서버 localhost(들)을 사용하여
본 사용자 이름과 암호가 필요합니다.
결과: 이 서버에서 완전한지 같은 방법(본인 연결 없이 리본 인증)
으로든 사용자 이름과 암호를 보내도록 증명하고 있습니다.

사용자 이름(U): admin
암호(P):

☐ 암호 저장(Y)

확인 취소

3 관리자 아이디와
패스워드를 입력하고
'확인' 버튼을 누르면
톰캣 관리자 프로그램의
화면이 나타납니다.

웹 애플리케이션의 목록

Apache Tomcat - Windows Internet Explorer
http://localhost:8080/

Administration
Status
Tomcat Manager
Documentation
Release Notes
Change Log

The Apache Software Foundation
http://www.apache.org/

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

`$CATALINA_HOME/webapps/ROOT/index.html`

where "\$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the README file.

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager". Users are defined in \$CATALINA_HOME/conf/tomcat-users.xml

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications. Tomcat mailing lists are available at the Tomcat project web site:

- users@tomcat.apache.org for general questions related to configuring and using Tomcat
- dev@tomcat.apache.org for developers working on Tomcat

Thanks for using Tomcat!

Powered by TOMCAT
Copyright © 1999-2010 Apache Software Foundation
All Rights Reserved

Manager - Windows Internet Explorer
http://localhost:8080/manager/html

The Apache Software Foundation
http://www.apache.org/

Tomcat Web Application Manager

Message: OK

Manager
List Applications HTML Manager Help Manager Help Server Status

Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	1	Start Stop Reload Undeploy Expire sessions with idle > 30 minutes
/servlet		true	1	Start Stop Reload Undeploy Expire sessions with idle > 30 minutes
/docs	Tomcat Documentation	true	1	Start Stop Reload Undeploy Expire sessions with idle > 30 minutes
/examples	Servlet and JSP Examples	true	1	Start Stop Reload Undeploy Expire sessions with idle > 30 minutes
/manager	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle > 30 minutes

- 톰캣의 아이디와 패스워드가 기억나지 않는다면 톰캣의 **conf** 디렉터리에 있는 **tomcat-users.xml** 파일을 열어서 확인한다.

3. 톰캣 관리자 프로그램 사용하기

❖ 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 **HTML**을 출력하는 **print, println, printf** 메서드 호출문보다 앞에 와야 할 뿐만 아니라, **response.getWriter** 메서드 호출문보다도 먼저 와야 한다.



3. 톰캣 관리자 프로그램 사용하기

❖ 한글 HTML 문서를 출력하는 서블릿 클래스

- 한글을 포함한 HTML 문서를 출력하는 서블릿 클래스는 다음과 같다.

[예제2-2] 1부터 100까지 더하는 서블릿 클래스

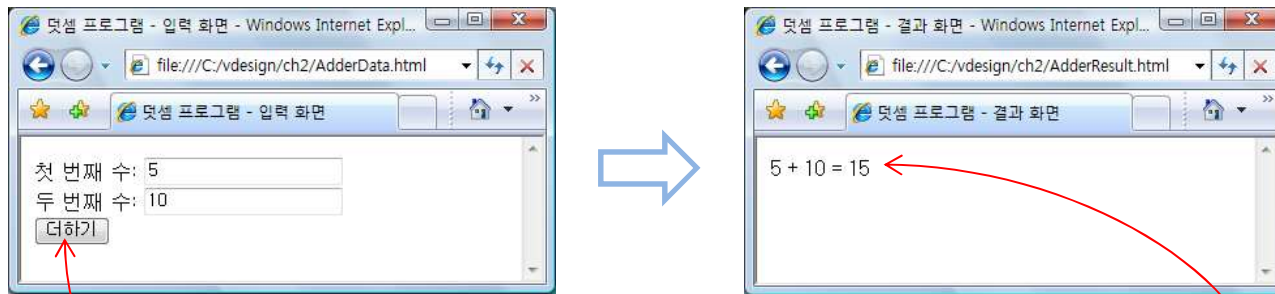
```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class HundredServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        int total = 0;
        for (int cnt = 1; 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> ");
    }
}
```



4. 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 보여주
는
웹 페이지가 나타난다.

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



4. 웹 브라우저로부터 데이터 입력받기

❖ 웹 브라우저로부터 데이터를 입력받는 서블릿 클래스

- 각 화면의 **URL**은 다음과 같이 정한다.

`http://localhost:8080/brain/AdderInput.html`

← [그림 2-21] 왼쪽 화면
URL

`http://localhost:8080/brain/adder`

← [그림 2-21] 오른쪽 화면
URL

- 왼쪽 화면은 **<FORM>** 엘리먼트를 사용해서 구현한다.

[예제2-3] 두 개의 수를 입력받는 HTML 문서

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type" content= "text/html; charset=euc-kr ">
    <TITLE>덧셈 프로그램 - 입력 화면</TITLE>
  </HEAD>
  <BODY>
    <FORM ACTION=/brain/adder>
      첫 번째 수: <INPUT TYPE=TEXT NAME=NUM1><BR>
      두 번째 수: <INPUT TYPE=TEXT NAME=NUM2><BR>
      <INPUT TYPE=SUBMIT VALUE= '더하기' >
    </FORM>
  </BODY>
</HTML>
```

위 문서를 **AdderInput.html**이라는 이름으로 **brain** 웹 애플리케이션 디렉터리에 저장한다.



4. 웹 브라우저로부터 데이터 입력받기

❖ 웹 브라우저로부터 데이터를 입력받는 서블릿 클래스

- 오른쪽 화면을 구현하는 서블릿 클래스는 [예제 2-3]을 통해 입력된 두 수를 받아서 합을 계산한 후 **HTML** 문서로 만들어서 출력해야 한다.
- **<FORM>** 엘리먼트를 통해 입력된 데이터는 **doGet, doPost** 메서드의 첫 번째 파라미터인 **HttpServletRequest** 타입의 파라미터에 대해 **getParameter** 메서드를 호출해서 가져올 수 있다.
 - 각 **<INPUT>** 서브엘리먼트를 통해 입력된 데이터를 가져오기 위해서는 다음과 같은 메서드를 호출해야 한다.

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

<INPUT> 엘리먼트의 NAME 애트리뷰트 값

- 이 메서드가 리턴하는 값은 수치 타입이 아니라 문자열 타입이다.



4. 웹 브라우저로부터 데이터 입력받기

- ❖ 웹 브라우저로부터 데이터를 입력받는 서블릿 클래스
 - 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 메
서드



4. 웹 브라우저로부터 데이터 입력받기

❖ 웹 브라우저로부터 데이터를 입력받는 서블릿 클래스

[예제2-4] 두 수의 합을 구하는 서블릿 클래스

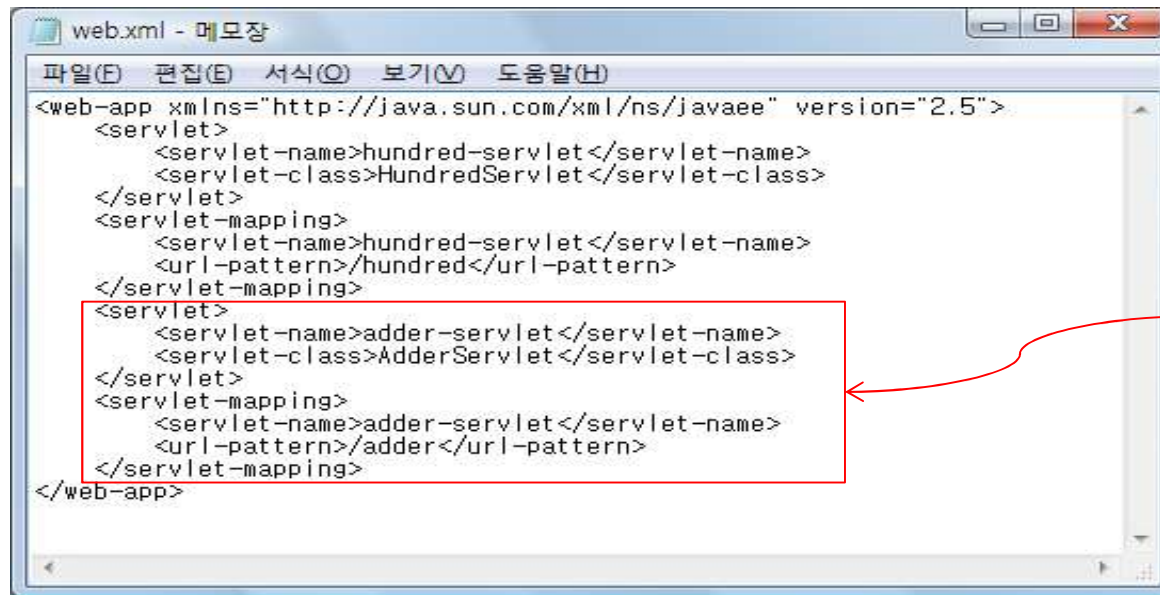
```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;

public class AdderServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        String str1 = request.getParameter( "NUM1 " );
        String str2 = request.getParameter( "NUM2 " );
        int num1 = Integer.parseInt(str1);
        int num2 = Integer.parseInt(str2);
        int sum = num1 + num2;
        response.setContentType( "text/html;charset=euc-kr " );
        PrintWriter out = response.getWriter();
        out.println( "<HTML> " );
        out.println( "<HEAD><TITLE>덧셈 프로그램 - 결과 화면</TITLE></HEAD> " );
        out.println( "<BODY> " );
        out.printf( "%d + %d = %d ", num1, num2, sum );
        out.println( "</BODY> " );
        out.println( "</HTML> " );
    }
}
```



4. 웹 브라우저로부터 데이터 입력받기

- ❖ 웹 브라우저로부터 데이터를 입력받는 서블릿 클래스
 - 서블릿 클래스 컴파일 후 그 결과를 **brain** 웹 애플리케이션 디렉터리의 **WEB-INF\classes** 서브디렉터리에 저장한다.
 - **WEB-INF** 디렉터리에 있는 **web.xml** 파일을 열어서 다음과 같이 서블릿 클래스를 등록한다.



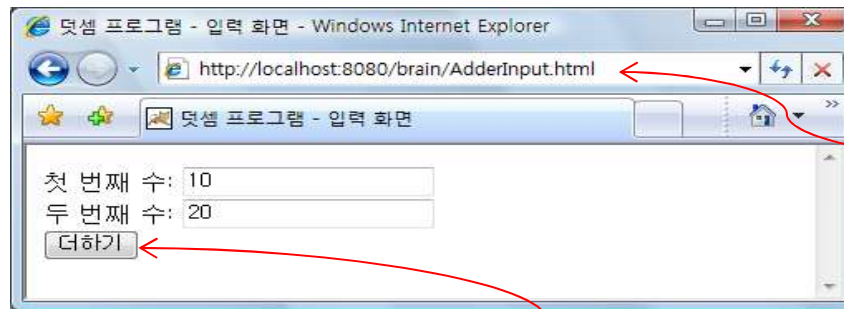
[예제2-4] 서블릿 클래스를 등록하는 코드

[그림 2-22] 예제 2-4의 서블릿 클래스를 등록하는 방법



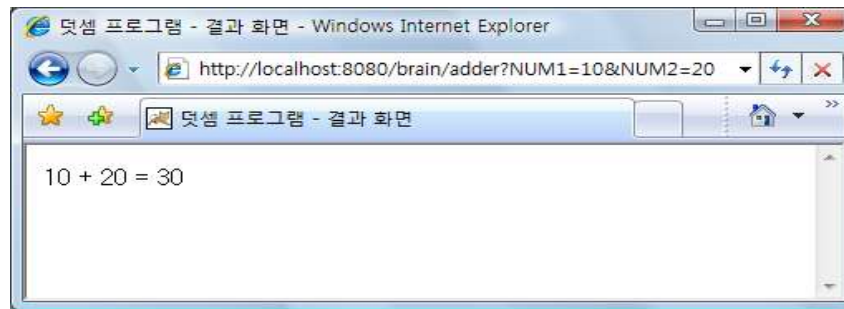
4. 웹 브라우저로부터 데이터 입력받기

- ❖ 웹 브라우저로부터 데이터를 입력받는 서블릿 클래스
 - 두 수의 합을 구하는 웹 애플리케이션의 실행 방법은 다음과 같다.



① [예제 2-3]의 URL을 입력한다.

② 두 수를 입력하고 더하기 버튼을 누르면
결과 화면이 나온다.



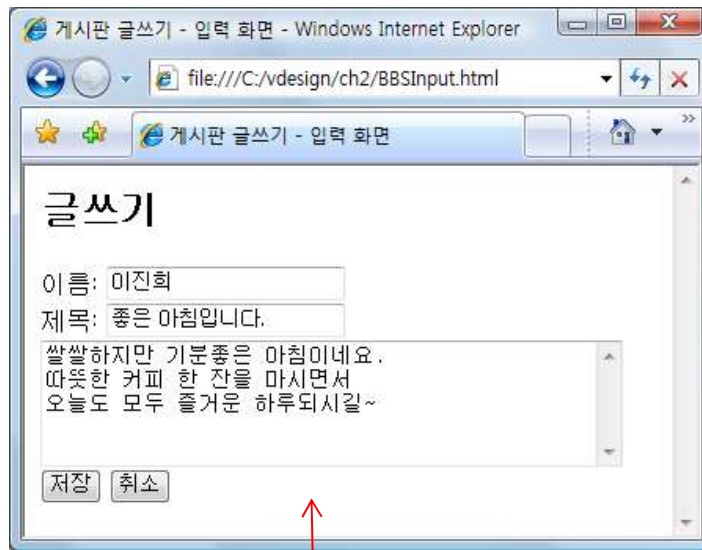
[그림 2-23] 예제 2-3, 예제 2-4의 실행 결과



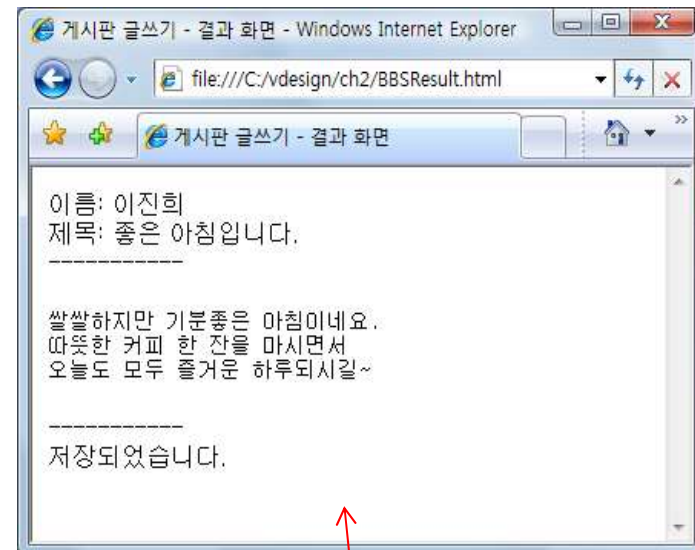
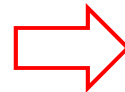
4. 웹 브라우저로부터 데이터 입력받기

❖ POST 메서드를 이용한 데이터 전송

- 웹 페이지를 통해 입력 받은 데이터를 웹 서버 쪽에 저장한 후에 또 다른 웹 페이지를 통해 저장된 결과를 보여주는 웹 애플리케이션이다.



① 이름, 제목, 내용을 입력하고 저장 버튼을 누르면



② 데이터가 웹 서버 쪽에 저장되고 결과 화면이 나타난다.

[그림 2-24] 게시판 글쓰기 애플리케이션 화면 설계



4. 웹 브라우저로부터 데이터 입력받기

❖ POST 메서드를 이용한 데이터 전송

- 입력 데이터가 클 경우에는 **URL** 뒷부분의 데이터가 잘려나갈 수 있으므로 **URL**이 아닌 별도의 영역을 통해 입력 데이터를 전송해야 한다.
- **<FORM>** 엘리먼트의 시작 태그에 **METHOD**라는 애트리뷰트를 추가하고, 애트리뷰트 값으로 **POST**를 지정하면 된다.

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

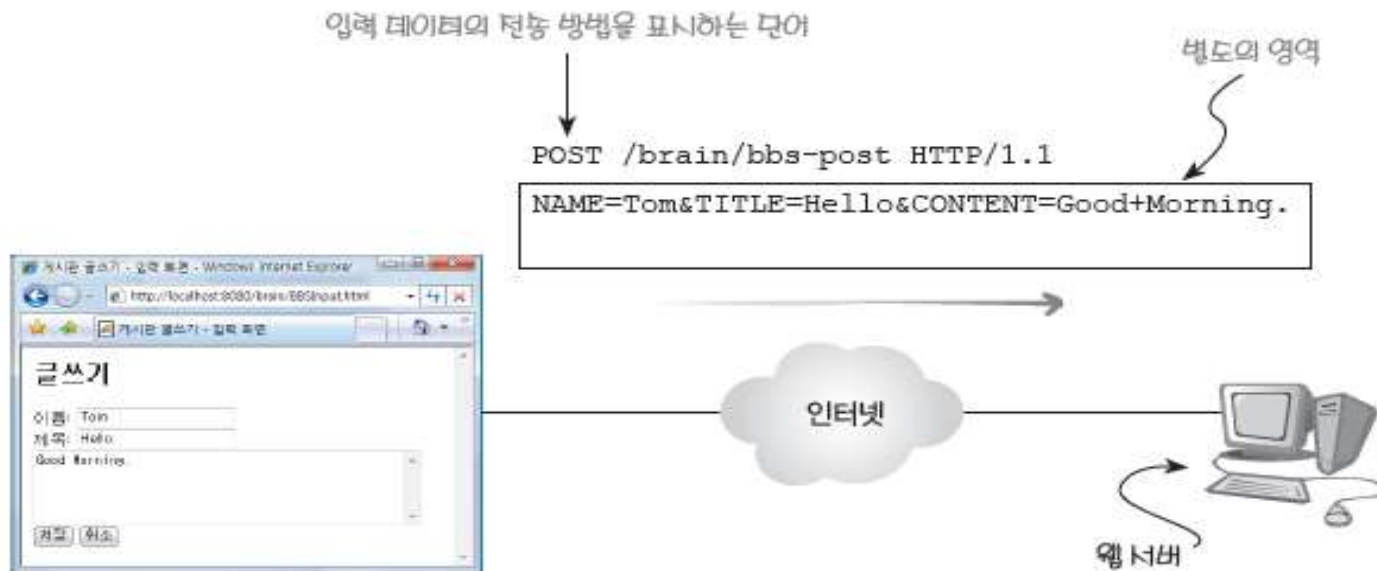
```
<FORM ACTION =/brain/bbs-post METHOD=POST>  
  이름 : <INPUT TYPE=TEXT NAME=WRITER>  
  제목 : <INPUT TYPE=TEXT NAME=TITLE>  
  <TEXTAREA NAME=CONTENT> </TEXTAREA>  
  <INPUT TYPE=SUBMIT VALUE= '저장' >  
  <INPUT TYPE=RESET VALUE= '취소' >  
</FORM>
```



4. 웹 브라우저로부터 데이터 입력받기

❖ POST 메서드를 이용한 데이터 전송

- <FORM> 엘리먼트를 통해 입력된 데이터는 URL 다음에 오는 별도의 영역을 통해 전송되며, URL 앞에는 POST라는 단어가 붙는다. 웹 서버는 이 단어를 보고 입력 데이터가 어디에 있는지 판단할 수 있다.



[그림 2-25] POST 메서드를 이용한 데이터 전송



4. 웹 브라우저로부터 데이터 입력받기

❖ POST 메서드를 이용한 데이터 전송

- URL을 정한 후 URL에 해당하는 HTML 문서를 작성한다.

`http://localhost:8080/brain/BBSInput.html`



[그림 2-24] 왼쪽 화면
URL

`http://localhost:8080/brain/bbs-post`



[그림 2-24] 오른쪽 화면
URL

- 첫 번째 화면을 구현하는 HTML 문서

[예제2-5] 게시판 글쓰기 기능의 데이터 입력을 위한 HTML 문서

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type" content= "text/html; charset=euc-kr" >
    <TITLE>게시판 글쓰기 - 입력 화면</TITLE>
  </HEAD>
  <BODY>
    <H2>글쓰기</H2>
    <FORM ACTION=/brain/bbs-post METHOD=POST>
      이름: <INPUT TYPE=TEXT NAME=NAME><BR>
      제목: <INPUT TYPE=TEXT NAME=TITLE><BR>
      <TEXTAREA COLS=50 ROWS=5 NAME=CONTENT></TEXTAREA><BR>
      <INPUT TYPE=SUBMIT VALUE= '저장' >
      <INPUT TYPE=RESET VALUE= '취소' >
    </FORM>
  </BODY>
</HTML>
```

- 이 예제를 톰캣의 `webapps\brain` 디렉터리에 `BBSInput.html`라는 이름으로 저장한다.



4. 웹 브라우저로부터 데이터 입력받기

❖ POST 메서드를 이용한 데이터 전송

■ 입력 데이터를 처리하는 서블릿 클래스의 작성 방법

- **doGet** 메서드를 선언하는 대신 **doPost** 메서드를 선언해야 한다. 웹 컨테이너는 **POST**라는 단어가 붙은 **URL**을 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.



4. 웹 브라우저로부터 데이터 입력받기

❖ POST 메서드를 이용한 데이터 전송

- 두 번째 화면을 구현하는 서블릿 클래스

[예제2-6] 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 name = request.getParameter( "NAME ");
        String title = request.getParameter( "TITLE ");
        String content = request.getParameter("CONTENT ");
        response.setContentType( "text/html;charset=euc-kr ");
        PrintWriter out = response.getWriter();
        out.println( "<HTML> ");
        out.println("<HEAD><TITLE>게시판 글쓰기 - 결과 화면</TITLE></HEAD> ");
        out.println( "<BODY> ");
        out.printf( "이름: %s <BR> ", name);
        out.printf( "제목: %s <BR> ", title);
        out.println( "-----<BR> ");
        out.printf( "<PRE>%s</PRE> ", content);
        out.println( "-----<BR> ");
        out.println( "저장되었습니다. ");
        out.println( "</BODY> ");
        out.println( "</HTML> ");

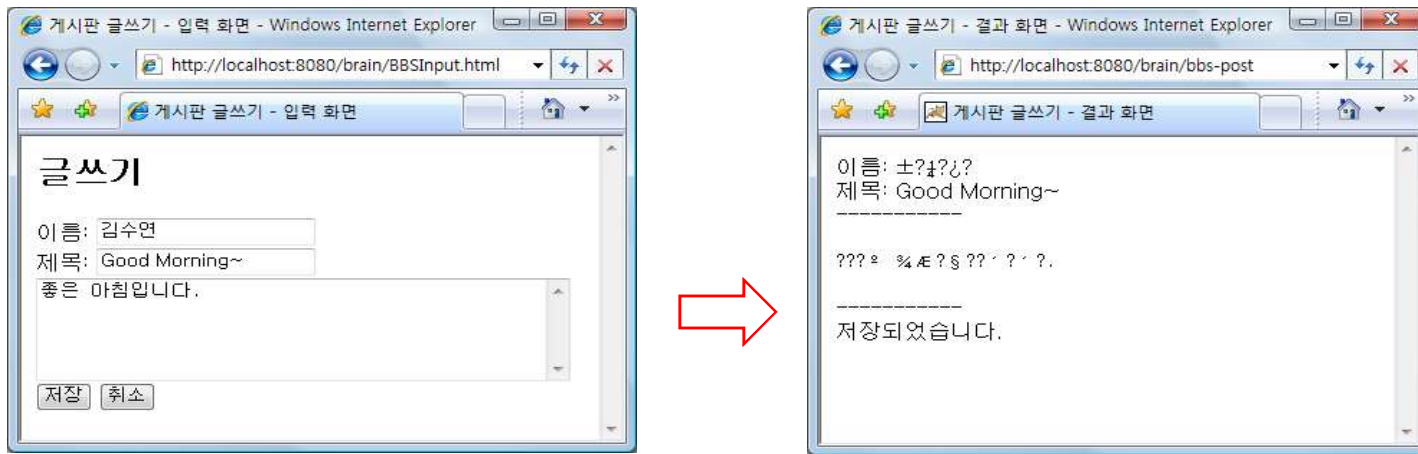
    }
}
```



4. 웹 브라우저로부터 데이터 입력받기

❖ POST 메서드를 이용한 데이터 전송

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



[그림 2-26] 예제 2-5, 예제 2-6의 실행 결과

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

```
request.setCharacterEncoding("euc-kr");
```

↑
한글 코드 이름



4. 웹 브라우저로부터 데이터 입력받기

❖ POST 메서드를 이용한 데이터 전송

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

[예제2-7] 게시판 글쓰기 기능을 처리하는 서블릿 클래스 - 완성

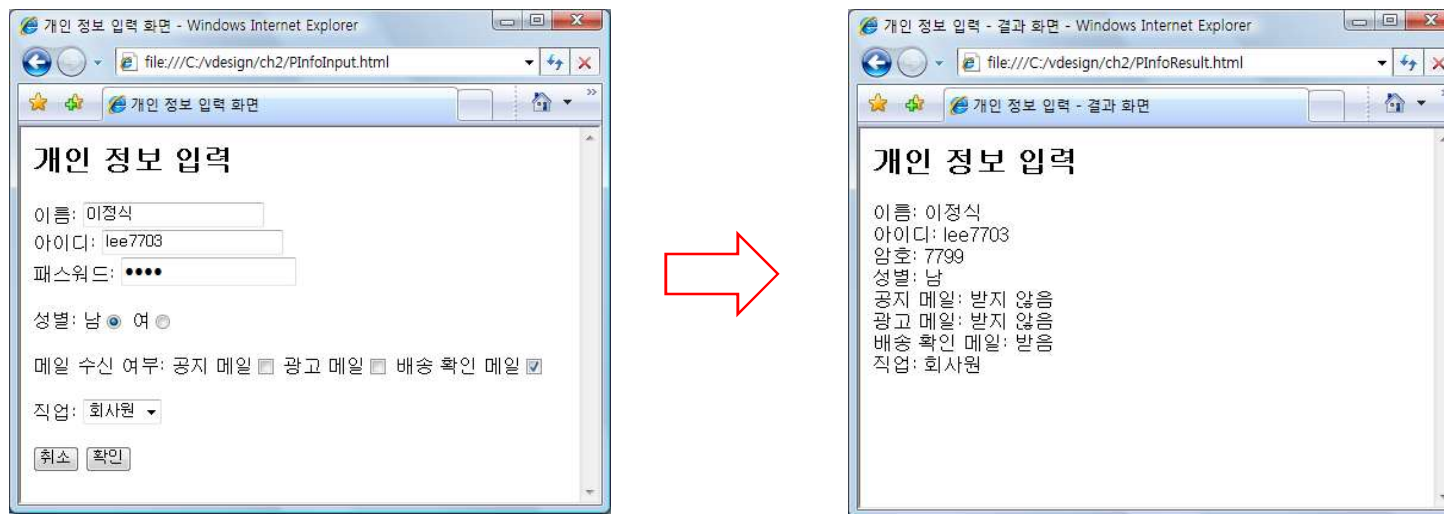
```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class BBSPostServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        request.setCharacterEncoding( "euc-kr ");
        String name = request.getParameter( "NAME " );
        String title = request.getParameter( "TITLE " );
        String content = request.getParameter( "CONTENT " );
        response.setContentType("text/html;charset=euc-kr ");
        PrintWriter out = response.getWriter();
        out.println( "<HTML> ");
        out.println( "<HEAD><TITLE>게시판 글쓰기 - 결과 화면</TITLE></HEAD> ");
        out.println( "<BODY> ");
        out.printf( "이름: %s <BR> ", name);
        out.printf( "제목: %s <BR> ", title);
        out.println( "-----<BR> ");
        out.printf( "<PRE>%s</PRE> ", content);
        out.println( "-----<BR> ");
        out.println( "저장되었습니다. ");
        out.println( "</BODY> ");
        out.println( "</HTML> ");
    }
}
```



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

- <FORM> 엘리먼트를 이용하면 텍스트 상자 외에도 라디오 버튼, 체크 박스, 선택 상자 등 다양한 형태로 데이터를 입력 받을 수 있다.



[그림 2-29] 개인 정보 입력 애플리케이션의 화면 설계

<http://localhost:8080/brain/PInfoInput.html>

<http://localhost:8080/brain/pinfo-result>

[그림2-29] 왼쪽 화면
URL

[그림2-29] 오른쪽 화면
URL



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

- 이름, 아이디 항목은 **<INPUT>** 엘리먼트의 **TYPE** 애트리뷰트 값을 **TEXT**로 지정해서 만들 수 있다. 패스워드는 **<INPUT>** 엘리먼트의 **TYPE** 애트리뷰트 값을 **PASSWORD**로 지정해서 만든다.

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

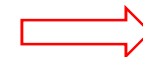


- PASSWORD** 타입으로 입력한 문자는 모니터상에 나타나지 않는다.
- 성별 항목은 라디오 버튼으로 만들어야 한다. 라디오 버튼은 **<INPUT>** 엘리먼트의 **TYPE** 애트리뷰트 값을 **RADIO**로 지정해서 만들 수 있으며, 반드시 **NAME**, **VALUE** 애트리뷰트를 써야 한다.

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

똑같은
NAME
애트리뷰트
값

각각 다른
VALUE
애트리뷰트 값



남 ☐ 여 ☒

한 항목을 선택하면 다른 항목의 선택이 해제된다.



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

- 메일 수신 여부 항목은 체크 박스로 만들어야 한다. 체크 박스는 **<INPUT>** 엘리먼트의 **VALUE** 속성을 **CHECKBOX**로 지정해서 만들 수 있으며, **NAME** 속성을 써야 한다. **NAME** 속성에는 각각 다른 값을 지정해야 한다.

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

각각 다른
VALUE
속성 값

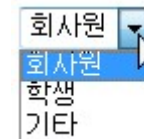


공지 메일 ☒ 광고 메일 ☐ 배송 확인 메일 ☒

한번 클릭하면 선택되고
또 한번 클릭하면 해제된
다.

- 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>
```



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

[예제2-8] 개인 정보를 입력받는 HTML 문서

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type " content= "text/html; charset=euc-kr ">
    <TITLE>개인 정보 입력 화면</TITLE>
  </HEAD>
  <BODY>
    <H2>개인 정보 입력</H2>
    <FORM ACTION=/brain/pinfo-result METHOD=GET>
      이름: <INPUT TYPE=TEXT NAME=NAME><BR>
      아이디: <INPUT TYPE=TEXT NAME=ID><BR>
      비밀번호: <INPUT TYPE=PASSWORD NAME=PASSWORD><BR><BR>
      성별:
        남<INPUT TYPE=RADIO NAME=GENDER VALUE=MALE>
        여<INPUT TYPE=RADIO NAME=GENDER VALUE=FEMALE><BR><BR>
      메일 수신 여부:
        공지 메일<INPUT TYPE=CHECKBOX NAME=INOTICE>
        광고 메일<INPUT TYPE=CHECKBOX NAME=CNOTICE>
        배송 확인 메일<INPUT TYPE=CHECKBOX NAME=DNOTICE><BR><BR>
      직업:
        <SELECT NAME=JOB>
          <OPTION>회사원</OPTION>
          <OPTION>학생</OPTION>
          <OPTION>기타</OPTION>
        </SELECT><BR><BR>
        <INPUT TYPE=RESET VALUE= "취소 ">
        <INPUT TYPE=SUBMIT VALUE= "확인 ">
    </FORM>
  </BODY>
</HTML>
```

- 이 예제를 톰캣의 **webapps\brain** 디렉터리에 **PInfoInput.html** 라는 이름으로 저장한다.



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

■ 서블릿 클래스의 작성 방법

- **<INPUT>** 엘리먼트의 **TYPE** 애트리뷰트 값이 **TEXT** 또는 **PASSWORD**일 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



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

- 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=FEMAIL>
```

```
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



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

- 선택 상자의 경우 **<SELECT>** 엘리먼트의 **NAME** 애트리뷰트 값을 **getParameter** 메서드에 넘겨줘야 한다.

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



```
String job = request.getParameter( "JOB " );
```

선택된 항목의
<OPTION> 엘리먼트 내
용

NAME 애트리뷰트
값



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

- [그림 2-29]의 오른쪽 화면을 구현하는 서블릿 클래스

[예제2-8] 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 password = request.getParameter( "PASSWORD " );
        String gender = request.getParameter( "GENDER " );
        if (gender.equals( "MALE " ))
            gender = "남 ";
        else
            gender = "여 ";
        String iNotice = request.getParameter( "INOTICE " );
        String cNotice = request.getParameter( "CNOTICE " );
        String dNotice = request.getParameter( "DNOTICE " );
        String job = request.getParameter( "JOB " );
        response.setContentType( "text/html;charset=euc-kr " );
        PrintWriter out = response.getWriter();
        out.println( "<HTML> " );
        out.println("<HEAD><TITLE>개인 정보 입력 - 결과 화면</TITLE></HEAD>");
```

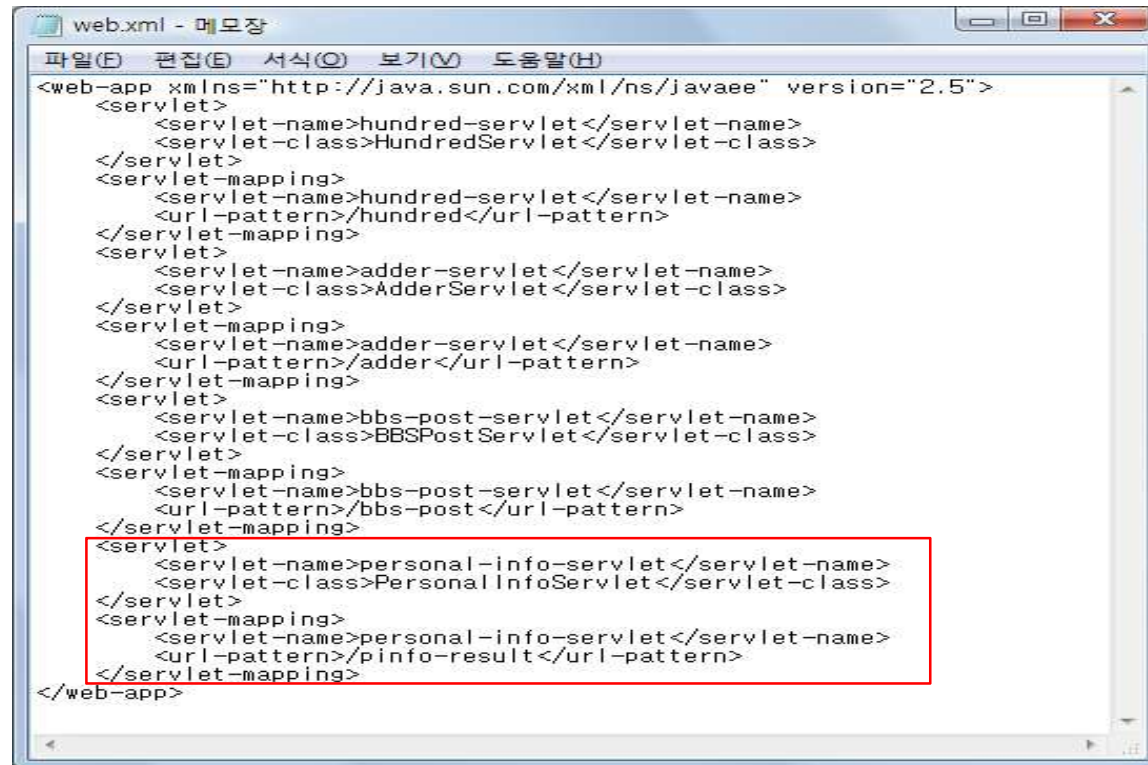
```
        out.println( "<BODY> " );
        out.println( "<H2>개인 정보 입력</H2> " );
        out.printf( "이름: %s <BR> ", name );
        out.printf( "아이디: %s <BR> ", id );
        out.printf( "암호: %s <BR> ", password );
        out.printf( "성별: %s <BR> ", gender );
        out.printf( "공지 메일: %s <BR> ", noticeToHangul(iNotice));
        out.printf( "광고 메일: %s <BR> ", noticeToHangul(cNotice));
        out.printf( "배송 확인 메일: %s <BR> ", noticeToHangul(dNotice));
        out.printf( "직업: %s <BR> ", job );
        out.println( "</BODY> " );
        out.println( "</HTML> " );
    }
    private String noticeToHangul(String notice) {
        if (notice == null)
            return "받지 않음 ";
        else if (notice.equals( "on " ))
            return "받음 ";
        else
            return notice;
    }
}
```



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

- 서블릿 클래스를 컴파일해서 결과를 톰캣의 `webapps\brain\WEB-INF\classes` 디렉터리에 저장한 후, **WEB-INF** 디렉터리의 `web.xml` 파일을 열어서 서블릿 클래스를 등록한다.



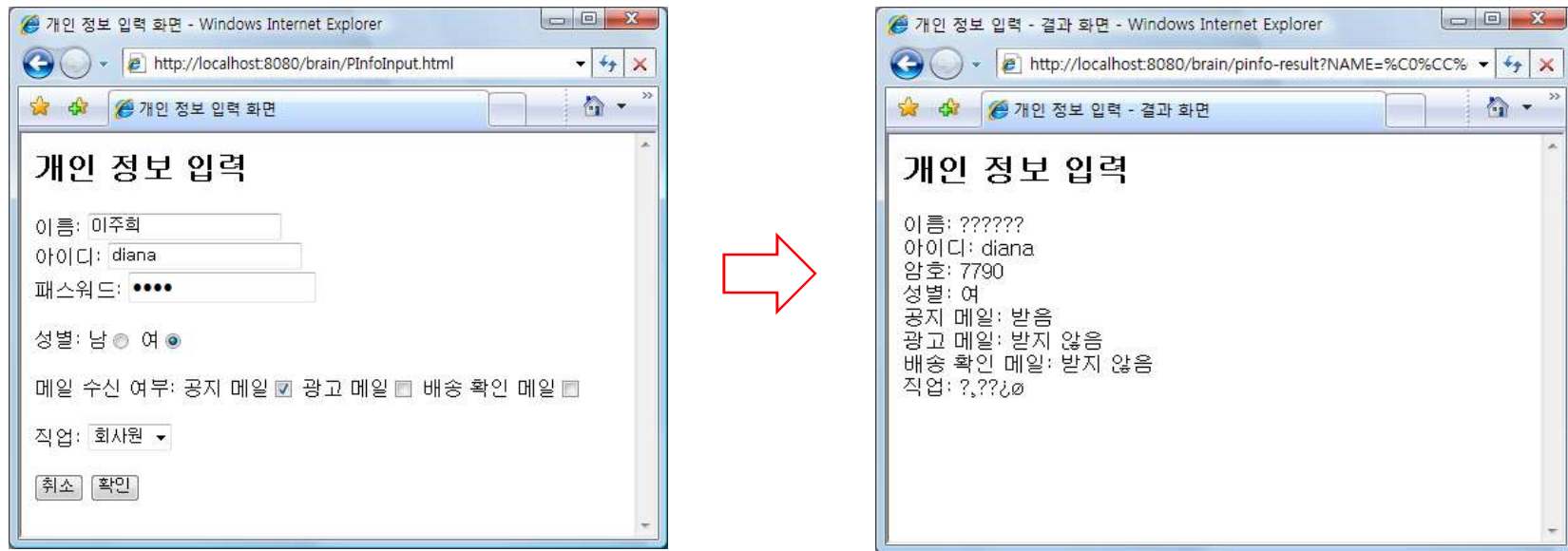
```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
  <servlet>
    <servlet-name>hundred-servlet</servlet-name>
    <servlet-class>HundredServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>hundred-servlet</servlet-name>
    <url-pattern>/hundred</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>adder-servlet</servlet-name>
    <servlet-class>AdderServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>adder-servlet</servlet-name>
    <url-pattern>/adder</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>bbs-post-servlet</servlet-name>
    <servlet-class>BBSPostServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>bbs-post-servlet</servlet-name>
    <url-pattern>/bbs-post</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>personal-info-servlet</servlet-name>
    <servlet-class>PersonalInfoServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>personal-info-servlet</servlet-name>
    <url-pattern>/pinfo-result</url-pattern>
  </servlet-mapping>
</web-app>
```



4. 웹 브라우저로부터 데이터 입력받기

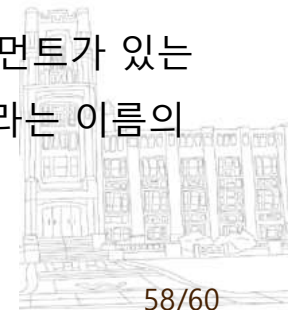
❖ 다양한 형태로 데이터 입력받기

- 다음 그림과 같이 한글 문제가 생길 경우



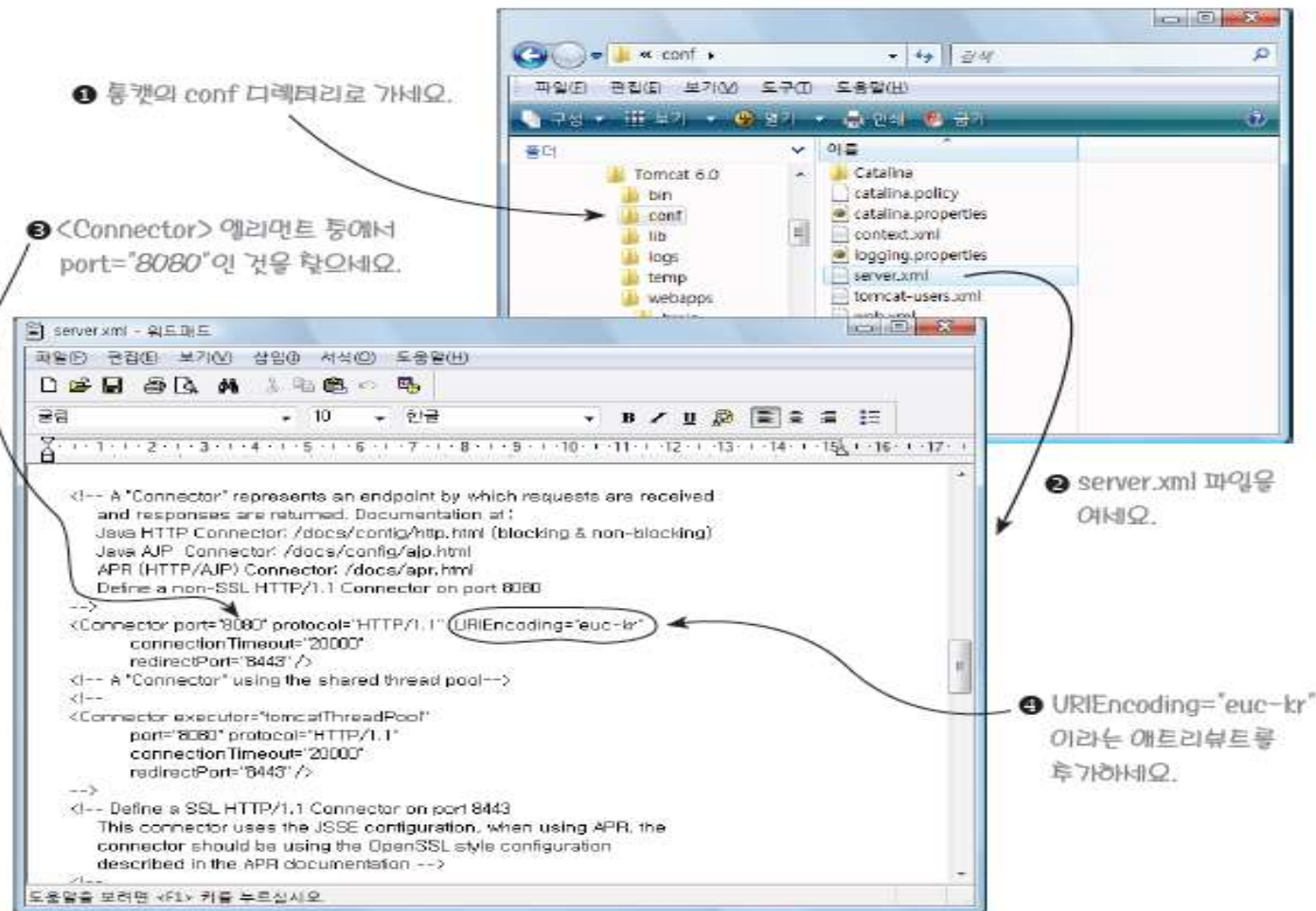
[그림 2-31] 예제 2-8, 2-9의 실행 결과 - 잘못된 결과

- 해결 방법: 톰캣의 **conf** 디렉터리에 있는 **server.xml** 파일은 열면 **<Connector>** 엘리먼트가 있는데, 그 중 **port** 애트리뷰트 값이 8080인 것을 찾아서 그 엘리먼트에 **URIEncoding**이라는 이름의 애트리뷰트를 추가하고 애트리뷰트 값으로 **"euc-kr"**를 지정한다.



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기



[그림 2-32] <FORM> 엘리먼트의 GET 메서드로 한글을 입력받기 위해 해야 할 일



4. 웹 브라우저로부터 데이터 입력받기

❖ 다양한 형태로 데이터 입력받기

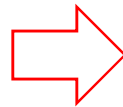
- **server.xml** 파일을 수정해서 저장하고, 톰캣을 재기동한 다음 예제를 다시 실행했을 때의 결과

개인 정보 입력 화면 - Windows Internet Explorer

http://localhost:8080/brain/pInfoInput.html

개인 정보 입력

이름: 김현수
아이디: kimhs77
패스워드:
성별: 남 ☐ 여 ☒
메일 수신 여부: 공지 메일 ☒ 광고 메일 ☒ 배송 확인 메일 ☒
직업: 학생
취소 확인



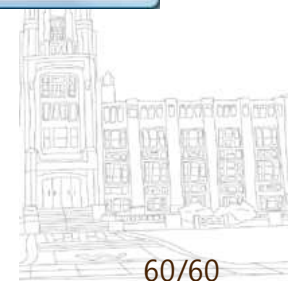
개인 정보 입력 - 결과 화면 - Windows Internet Explorer

http://localhost:8080/brain/pinfo-result?NAME=%B1%E8%

개인 정보 입력

이름: 김현수
아이디: kimhs77
암호: 9876
성별: 여
공지 메일: 받음
광고 메일: 받지 않음
배송 확인 메일: 받음
직업: 학생

[그림 2-33] 예제 2-8, 2-9의 실행 결과 - 올바른 결과





Thank You !

뇌를 자극하는 JSP & Servlet