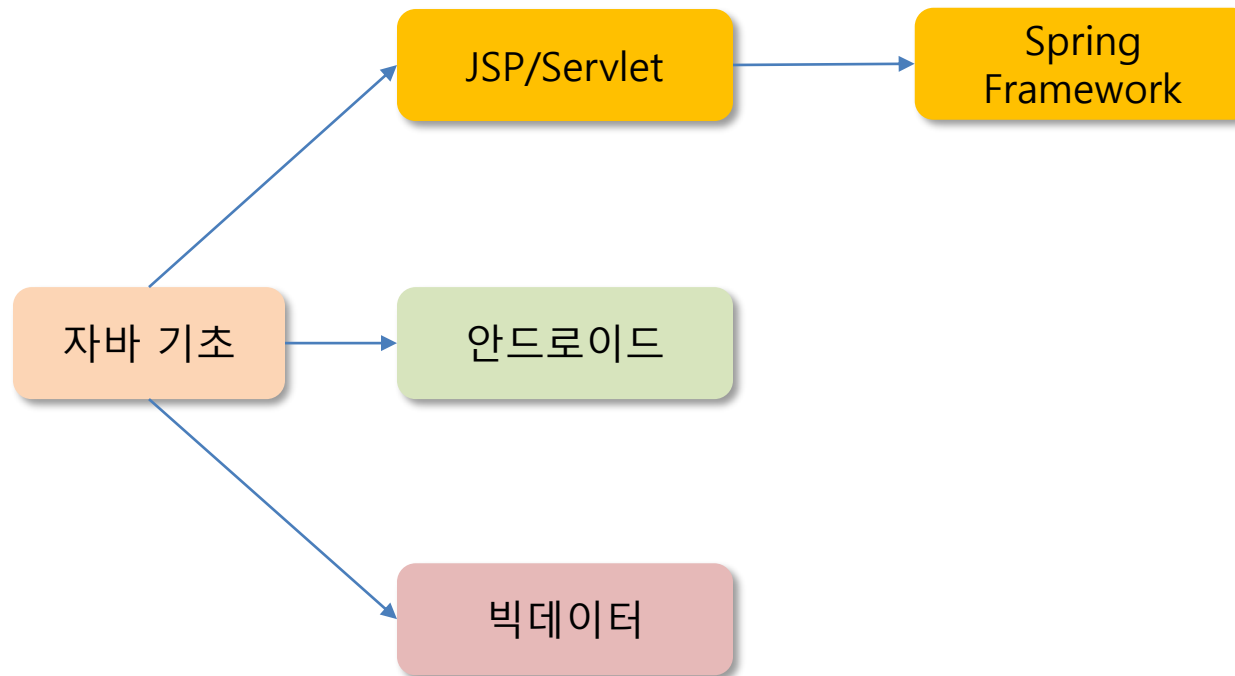
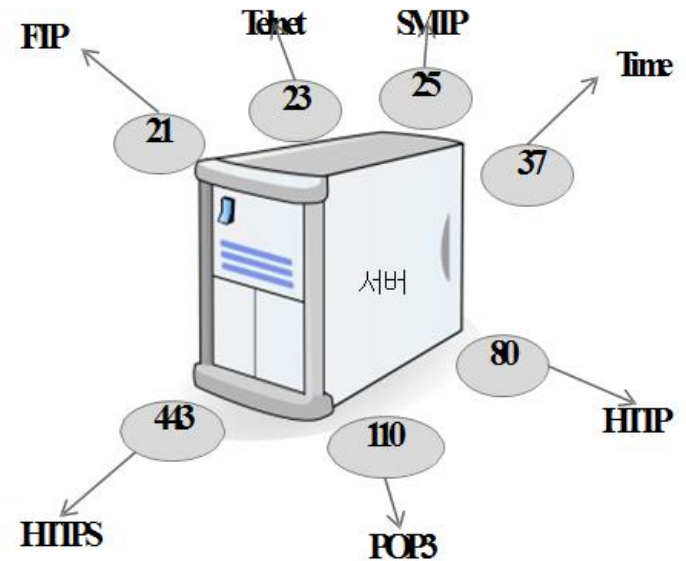
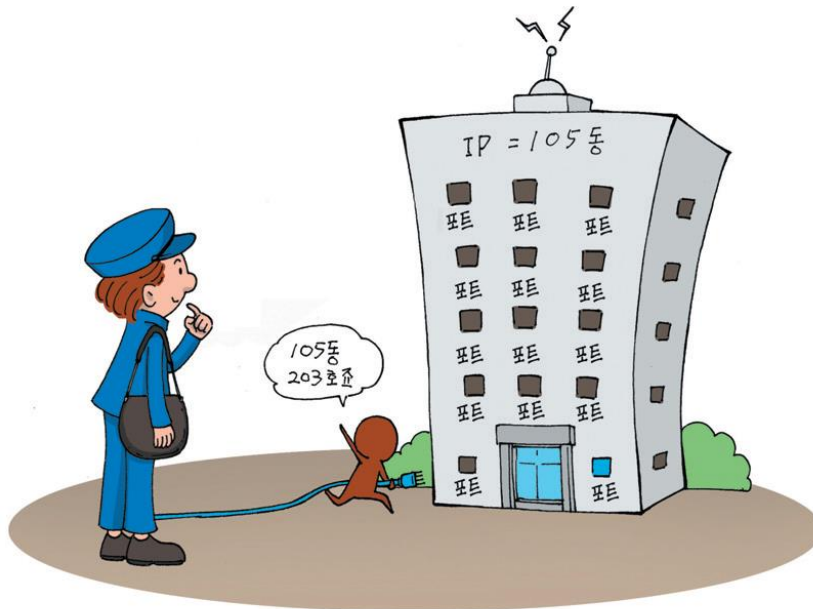


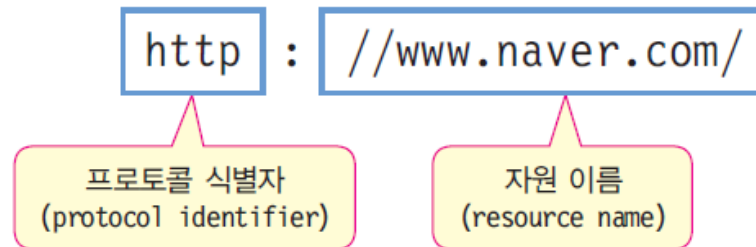
JSP



- ip 주소
 - 인터넷상에서 하나의 컴퓨터를 찾아갈수 있는 값
 - 아파트의동에 해당
- 포트
 - 인터넷상의 하나의 컴퓨터 내에서 프로그램을 찾아갈수 있는 값
 - 아파트의 호수에 해당

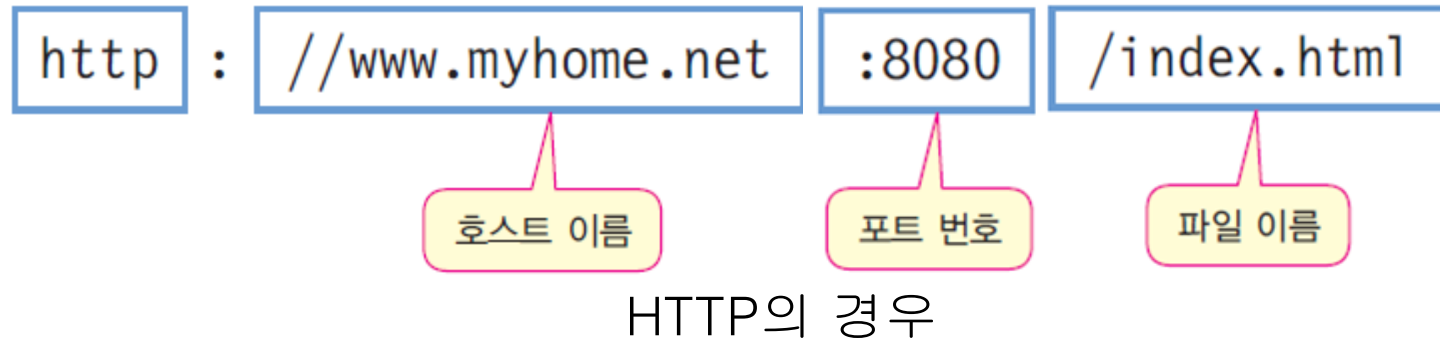


- URL이란?
 - URL은 Uniform Resource Locator
 - 인터넷 상의 리소스에 대한 주소
- URL 구조

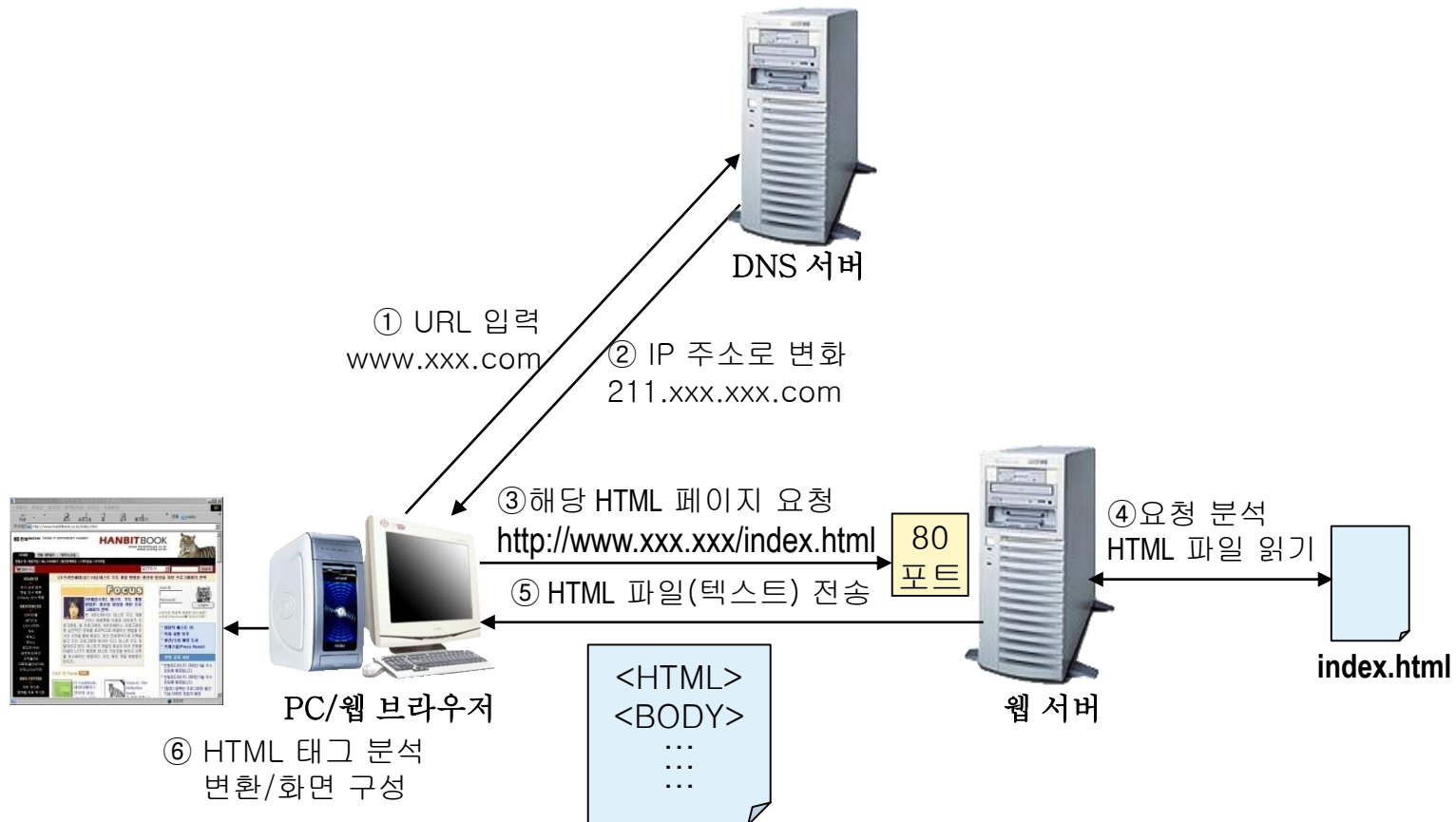


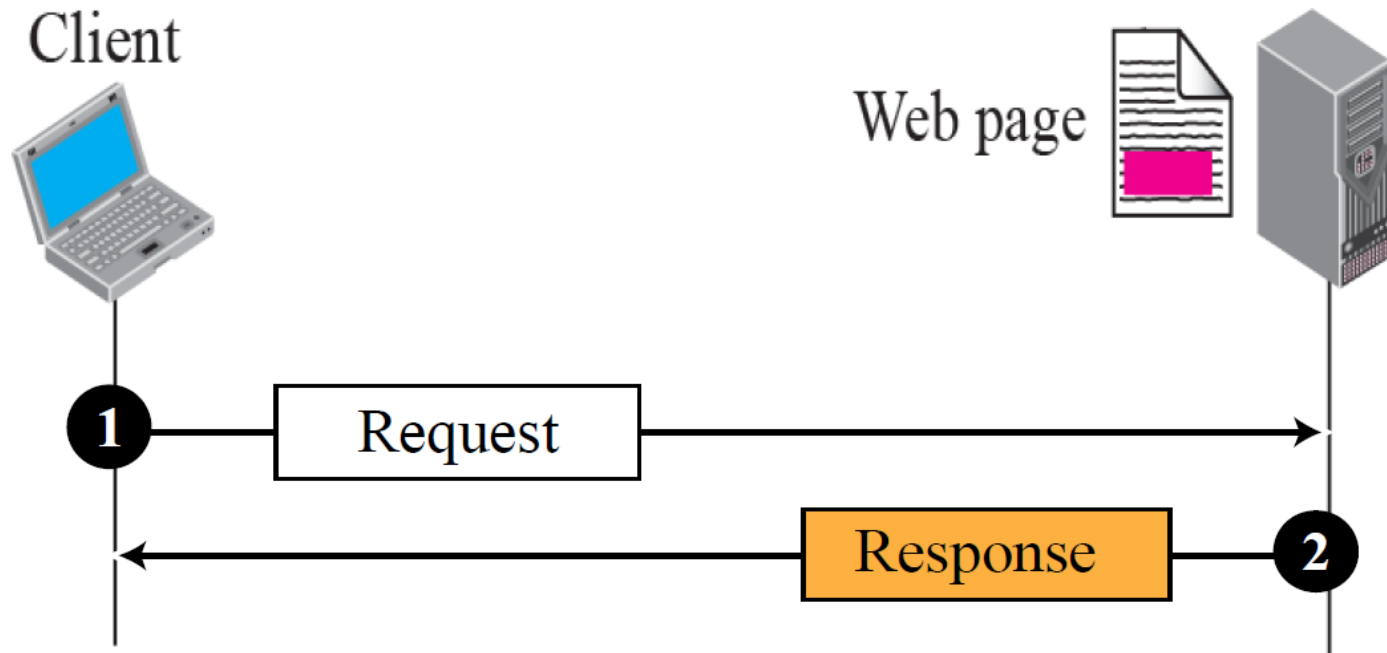
- Http
 - Hypertext Transfer Protocol
 - World Wide Web에 접속 하기 위한 프로토콜

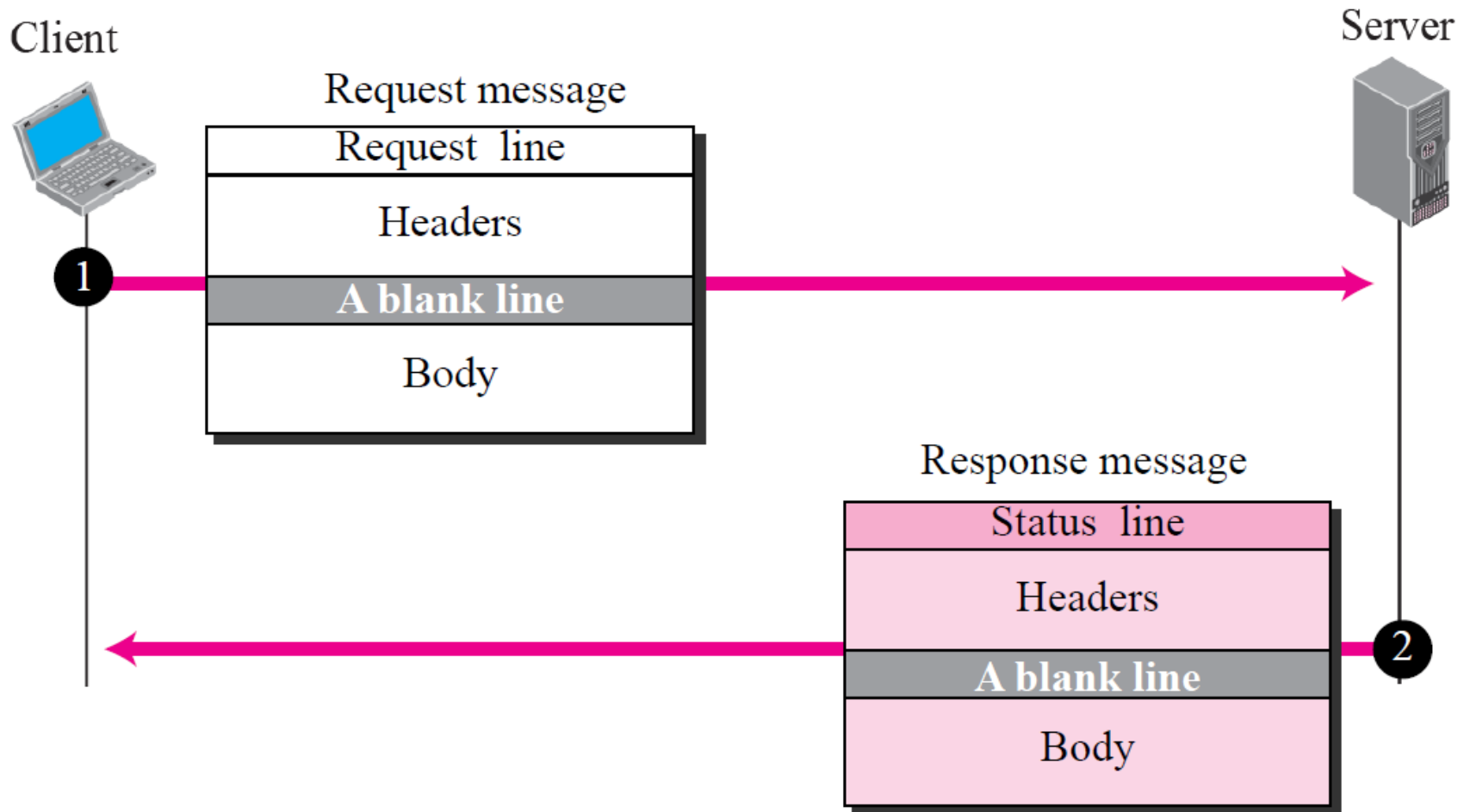
- 자원 이름
 - 자원 이름은 사용되는 프로토콜에 따라서 그 구성이 달라짐



- 일반적인 WWW서비스의 동작 과정





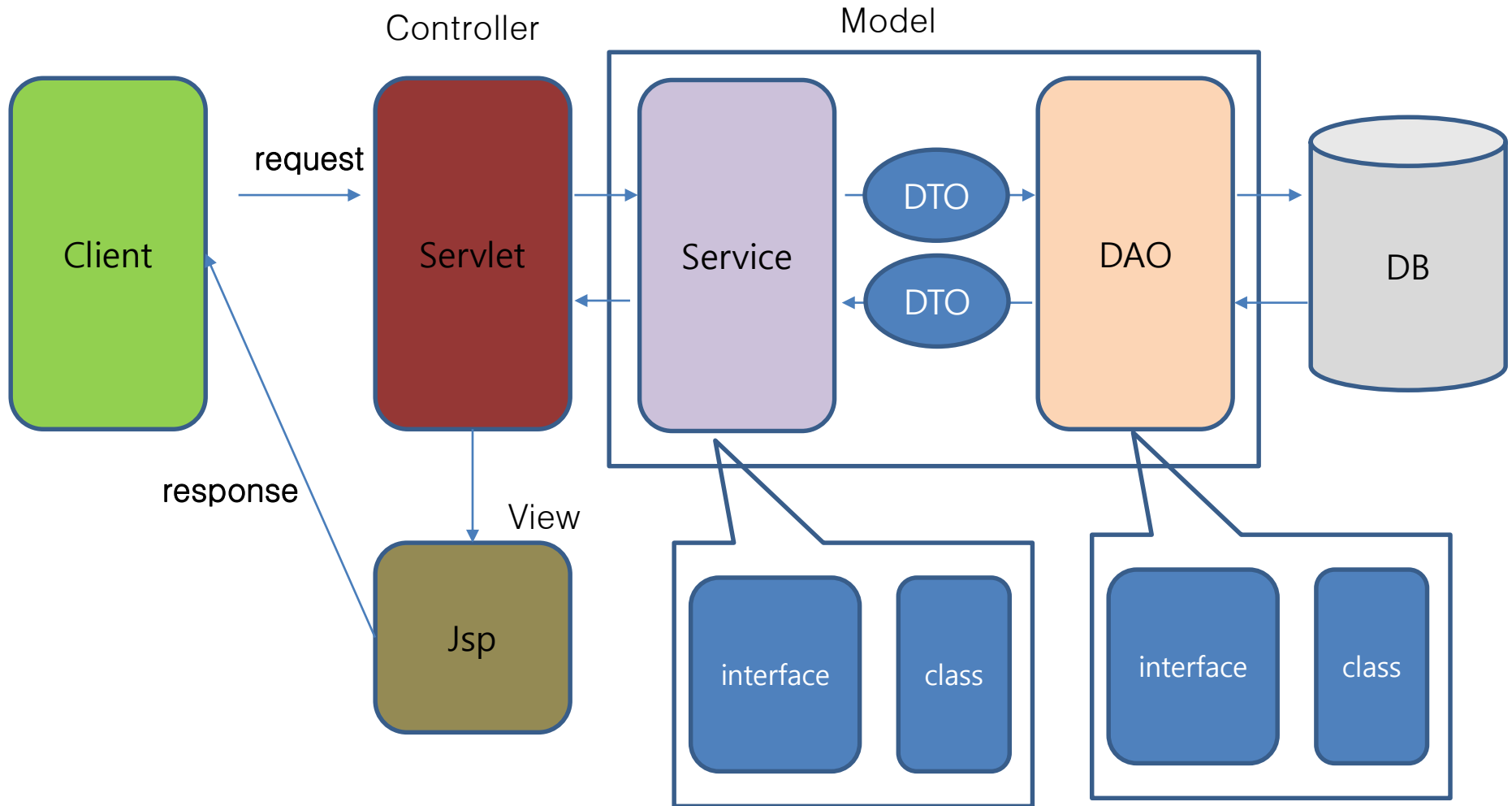


- GET 방식

- 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

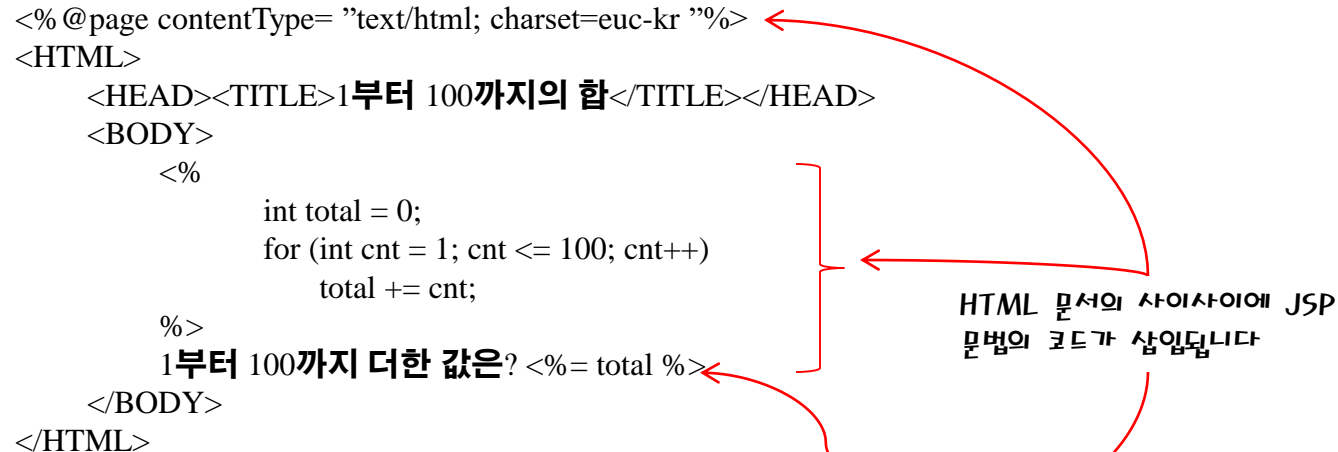
- 서버로 정보를 올리기 위해 설계된 방법이다. 예를 들어 HTML 폼에 입력한 내용을 서버에 전달하기 위한 요청.
- 서버에 전달 할 수 있는 데이터 크기에는 제한이 없다.
- URL에는 매개변수가 표시되지 않는다.



- JSP 기술에서 웹 애플리케이션을 구현할 때 작성하는 코드를 JSP 페이지라고 한다.
- JSP 페이지는 HTML 문서의 사이에 JSP 문법의 코드가 삽입되는 형태로 작성된다.

```
<%@page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
  <BODY>
    <%
      int total = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
        total += cnt;
    %>
    1부터 100까지 더한 값은? <%= total %>
  </BODY>
</HTML>
```

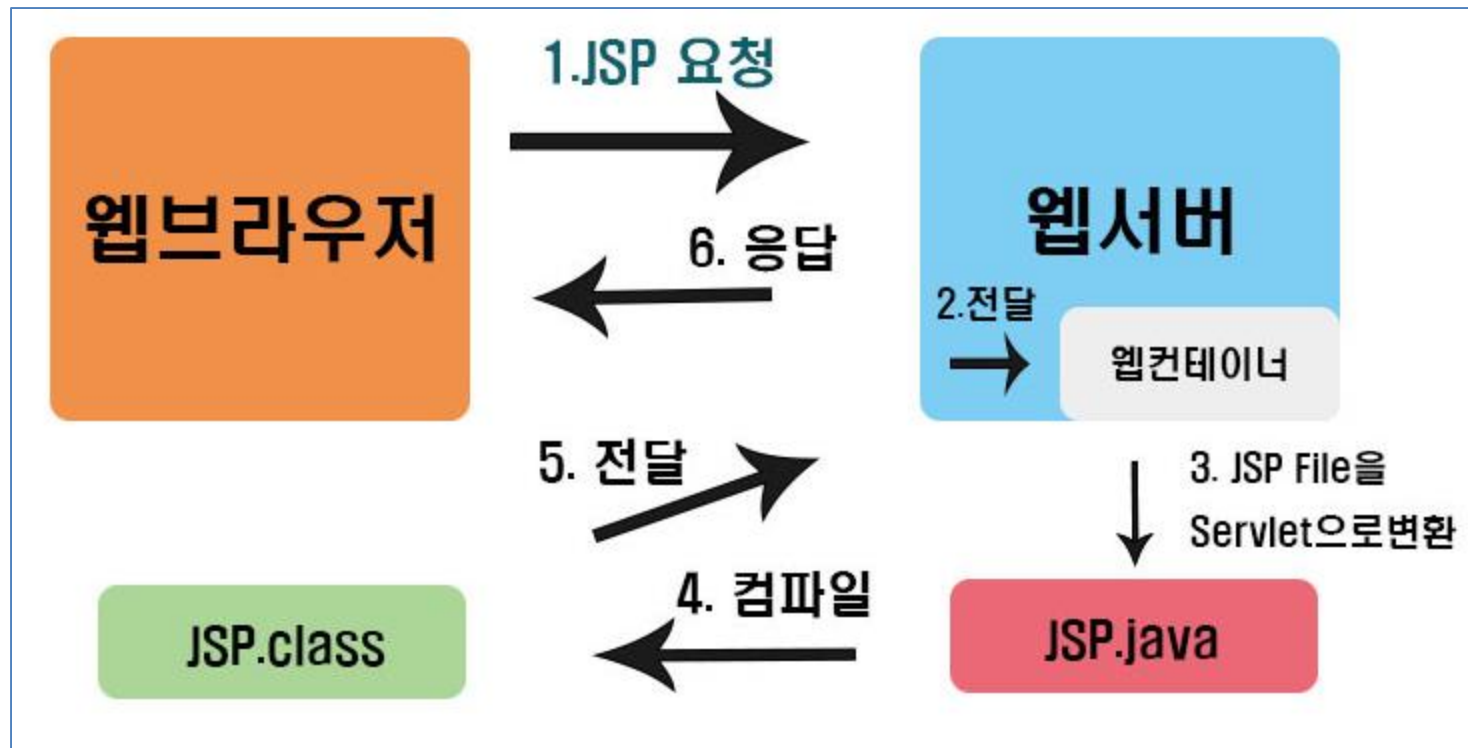
HTML 문서의 사이사이에 JSP 문법의 코드가 삽입됩니다



[그림 3-1] JSP 페이지의 형태

- JSP 페이지에 있는 HTML 코드는 웹 브라우저로 그대로 전송되지만, JSP 문법의 코드는 웹 컨테이너 쪽에서 실행되고 그 결과만 웹 브라우저로 전송된다.

1. JSP 처리 과정



- **java project에서 1부터 100까지의 합을 구하고 출력하는 코드 작성**

- **스크립팅 요소의 문법**
 - 스크립팅 요소(scripting elements)란 다음 세가지 문법을 말한다.
 - 선언부(declaration)
 - 스크립틀릿(scriptlet)
 - 익스프레션(expression)

■ 스크립틀릿(scriptlet)

- <%로 시작해서 %> 로 끝남
- 그 사이 자바 코드가 들어감
- 이 명령문은 웹 브라우저로 전송되는 것이 아니라 웹 서버 쪽에 실행 됨

JSP 페이지의 코드

```
<HTML>
  <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
  <BODY>
    <%
      int total = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
        total += cnt;
    %>
  </BODY>
</HTML>
```

■ 익스프레션(expression)

- <%=로 시작해서 %>로 끝남
- html 태그없이 다른 메소드 도움없이 출력됨
- 상수나 변수 선언(지역변수) , 연산식, 리턴 값이 있는 메서드 호출식이 들어갈 수 있음
- **실행문 뒤에 ; 없음에 주의 !!**

<%= total %>

자바 식

<%= total + 101 %>

자바 식

<%= Math.sqrt(num) %>

자바 식

JSP 페이지의 코드

```
<HTML>
  <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
  <BODY>
    <%
      int total = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
        total += cnt;
    %>
    1부터 100까지 더한 값은? <%= total %>
  </BODY>
</HTML>
```


- JSP 페이지의 한 스크립틀릿 안에서 선언한 변수를 그 뒤에 나오는 다른 스크립틀릿 안에 사용하는 것이 가능하다.

[예제3-1] 여러 개의 스크립틀릿이 있는 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>1부터 200까지의 합</TITLE></HEAD>
  <BODY>
    <%
      int total = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
        total += cnt;
    %>
    1부터 100까지의 합 = <%= total %> <BR>
    <%
      for (int cnt = 101; cnt <= 200; cnt++)
        total += cnt;
    %>
    1부터 200까지의 합 = <%= total %> <BR>
  </BODY>
</HTML>
```

total 변수를 선언한다

total 변수를 사용한다

■ 선언부(declaration)

- <%!로 시작해서 %>로 끝남
- 변수 선언(멤버변수)이나 메서드 선언문을 쓸수 있음.
- final, public, private, protected, static 등의 키워드를 붙이는 것도 가능

```
<%! final static int MAX=10000; %>
```

변수 선언

```
<%!
```

```
private int add(int num1, int num2) {  
    int sum = num1 + num2;  
    return sum;  
}
```

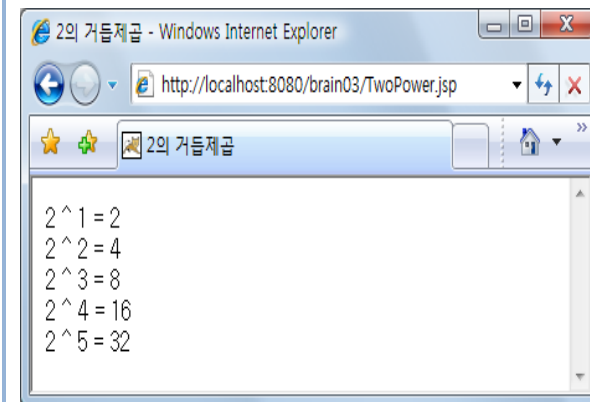
```
%>
```

메서드 선언

[예제3-2] 선언부를 포함한 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>2의 거듭제곱</TITLE></HEAD>
  <BODY>
    2 ^ 1 = <%= power(2, 1) %> <BR>
    2 ^ 2 = <%= power(2, 2) %> <BR>
    2 ^ 3 = <%= power(2, 3) %> <BR>
    2 ^ 4 = <%= power(2, 4) %> <BR>
    2 ^ 5 = <%= power(2, 5) %> <BR>
  </BODY>
</HTML>
<%!
  private int power(int base, int exponent) {
    int result= 1;
    for (int cnt = 0; cnt < exponent; cnt++)
      result *= base;
    return result;
  }
%>
```

선언부



- 아래와 같은 기능을 jsp로 만드세요
 - 두개의 재료값은 스크립틀릿 에서 지역 변수로 선언 사용
 - +, -, *, / 의 4개의 메소드를 선언 구현
 - 매개변수 : double 2개
 - 리턴타입 : double
 - 실행 결과 값은 expression

- **지시자의 문법**

- 지시자(directive)는 JSP의 다른 문법들(스크립팅 요소, 익스프레션 언어, 액션)과는 다른 목적으로 사용된다.
- 웹 브라우저로부터의 요청을 처리하는 것이 아니라, 웹 컨테이너가 JSP 페이지를 서블릿 클래스로 변환할 때 필요한 정보들을 기술하는 역할을 한다.
- JSP 페이지에 사용할 수 있는 지시자의 종류
 - page 지시자
 - include 지시자
 - taglib 지시자

- **지시자의 문법**

- <%@으로 시작하고 %>로 끝남
- page 지시자는 JSP 페이지 전체에 적용되는 정보를 기술하기 위해 사용된다.

애트리뷰트 이름	기술하는 정보/애트리뷰트의 역할	다루는 장
contentType	JSP 페이지가 생성하는 문서의 종류와 그 문서를 웹 브라우저로 전송할 때 사용되는 인코딩 타입	3장
import	스크립팅 요소 안에서 사용할 자바 클래스와 인터페이스를 임포트하기 위해 사용하는 애트리뷰트	3장
buffer	출력 버퍼의 크기	3장
autoFlush	출력 버퍼가 모두 찼을 때의 동작	3장
isThreadSafe	JSP 페이지가 싱글-스레드 모드로 작동하도록 만들기 위해 필요한 애트리뷰트	3장
session	JSP 페이지의 세션 참여 여부	4장
errorPage	에러를 처리할 JSP 페이지의 URL	5장
isErrorPage	에러를 처리하는 JSP 페이지인지 여부	부록 A
isELIgnored	익스프레션 언어의 무시/처리 여부	다루지 않음
pageEncoding	JSP 페이지의 인코딩 타입	다루지 않음
info	JSP 페이지에 대한 설명	다루지 않음
extends	JSP 페이지로부터 생성되는 서블릿 클래스의 슈퍼클래스	다루지 않음
language	스크립팅 요소 안에서 사용할 프로그래밍 언어. 현재는 ‘java’ 라는 값만 지정할 수 있음	다루지 않음
deferredSyntaxAllowedAsLiteral	익스프레션 언어의 예약 문자열인 ‘#{’ 를 사용했을 때의 에러 발생 여부	다루지 않음
trimDirectiveWhitespaces	지시자 바로 다음에 있는 공백 문자를 제거하기 위해 사용하는 애트리뷰트	다루지 않음

[표 3-1] page 지시자의 애트리뷰트

- page 지시자

- contentType 애트리뷰트는 JSP 페이지가 생성하는 문서의 종류와 그 문서를 웹 브라우저로 전송할 때 사용할 인코딩 방식을 지정하기 위해 사용한다. 이 두 값은 세미콜론(;)으로 구분해서 써야 한다.

```
<%@page contentType= "text/html; charset=euc-kr" %>
```

한글이 포함된 JSP 페이지일 경우

```
<%@page contentType= "text/html" %>
```

ASCII 코드로만 구성된 JSP 페이지일 경우

↑
'text/html' 은 contentType 애트리뷰트의 디폴트 값이므로 이 page 지시자는 생략할 수 있다

■ page 지시자의 import 애트리뷰트

- 자바의 import 문과 마찬가지로 다른 패키지에 속하는 클래스나 인터페이스를 임포트하는 역할을 한다.

```
<%@page import="java.util.GregorianCalendar" %>
```

java.util 패키지의 GregorianCalendar 클래스를 임포트한다.

```
<%@page import="java.util.*" %>
```

java.util 패키지의 모든 클래스와 인터페이스를 임포트한다

```
<%@page import="java.util.ArrayList, java.io.*" %>
```

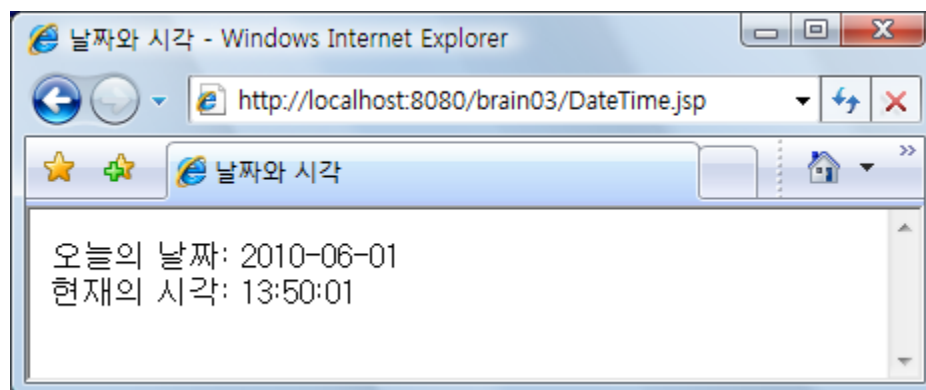
java.util.ArrayList 클래스와 java.io 패키지의 모든 클래스, 인터페이스를 임포트한다

- import 애트리뷰트의 사용 예를 보여주는 JSP 페이지

[예제3-3] page 지시자의 import 애트리뷰트 사용 예

```
<% @page contentType= "text/html; charset=euc-kr" %>
<% @page import= "java.util.GregorianCalendar" %>
<HTML>
  <HEAD><TITLE>날짜와 시각</TITLE></HEAD>
  <BODY>
    <%
      GregorianCalendar now = new GregorianCalendar();
      String date = String.format( "%TF ", now);
      String time = String.format( "%TT ", now);

      %>
      오늘의 날짜: <%= date %> <BR>
      현재의 시각: <%= time %> <BR>
    </BODY>
  </HTML>
```



- 다음을 jsp 코드로 만드세요
 - 2021.04.09 14:28:12 형태의 값이 브라우저에 출력 되어야 함

2021.04.09 14:28:12 형태로 현재 시간 출력하기

```
import java.util.*;
import java.text.*;
public class MainTime {

    public static void main(String[] args) {
        SimpleDateFormat formatter = new SimpleDateFormat ( "yyyy.MM.dd HH:mm:ss", Locale.KOREA );
        Date currentTime = new Date ( );
        String dTime = formatter.format ( currentTime );
        System.out.println ( dTime );

    }

}
```

- **include 지시자**

- 다른 JSP 페이지 또는 HTML 문서를 불러다가 현재 JSP 페이지의 일부로 만들기 위해 사용한다.
- 불러올 대상은 file 애트리뷰트를 이용해서 지정할 수 있으며, 이 애트리뷰트의 값은 지시자가 속하는 JSP 페이지를 기준으로 한 상대적인 URL로 해석된다.

```
<%@include file= "Today.jsp" %>
```

현재 디렉터리에 있는 Today.jsp를
include한다

```
<%@include file= "sub1/Today.jsp" %>
```

sub1 디렉터리에 있는 Today.jsp를
include한다

[예제3-4] include 지시자의 사용 예

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>오늘의 메뉴</TITLE></HEAD>
  <BODY>
    <H3>오늘의 메뉴</H3>
    - 삼계탕 <BR>
    - 돈까스 <BR>
    - 튀김국수 <BR><BR>
    <%@include file= "Today.jsp" %>
  </BODY>
</HTML>
```



Today.jsp를 include한다

Today.jsp

```
<body>
  today set menu
  - 빅맥 <BR>
  - 버거킹 <BR>
</body>
```

오늘의 메뉴

- 삼계탕
- 돈까스
- 튀김국수

today set menu - 빅맥
- 버거킹

- 지시자의 문법

- taglib 지시자는 JSP 문법 중 하나인 액션(action)을 사용할 때 필요하다.
- taglib 지시자는 액션이 속한 라이브러리를 설치해야만 사용할 수 있다.

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

액션이 속하는 라이브러리를 지정하는 지시자

- 주석을 기술하는 방법

- JSP 페이지에 주석을 다는 방법은 다양하다.
- JSP 페이지의 HTML 코드 부분 : `<!--`로 시작해서 `-->`로 끝나는 HTML 주석을 쓸 수 있다.

`<!-- HTML의 주석 -->`

↑ 시작 표시 ↑ 끝 표시

- JSP 페이지의 스크립팅 요소 안 : 자바 문법을 따르는 주석을 쓸 수 있다.

`/* Java의 주석 */`

↑ 시작 표시 ↑ 끝 표시

`// Java의 주석`

↑ 시작 표시

- JSP 고유의 주석을 사용할 수 있다.

`<%-- JSP의 주석 --%>`

↑ 시작 표시 ↑ 끝 표시

주석을 기술하는 방법

여러 가지 주석의 사용 예를 보여주는 JSP 페이지

[예제3-5] 여러 가지 주석을 포함하는 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" %>

<HTML>

  <HEAD><TITLE>1부터 10까지의 곱</TITLE></HEAD>

  <!-- 이것은 JSP에 의해 생성된 HTML 문서입니다. -->

  <BODY>

    <%-- 다음은 데이터를 처리하는 스크립틀릿입니다. --%>

    <% int result = 1; // 곱을 저장할 변수

      /* 1부터 10까지 곱하는 반복문 */

      for (int cnt = 1; cnt <= 10; cnt++)

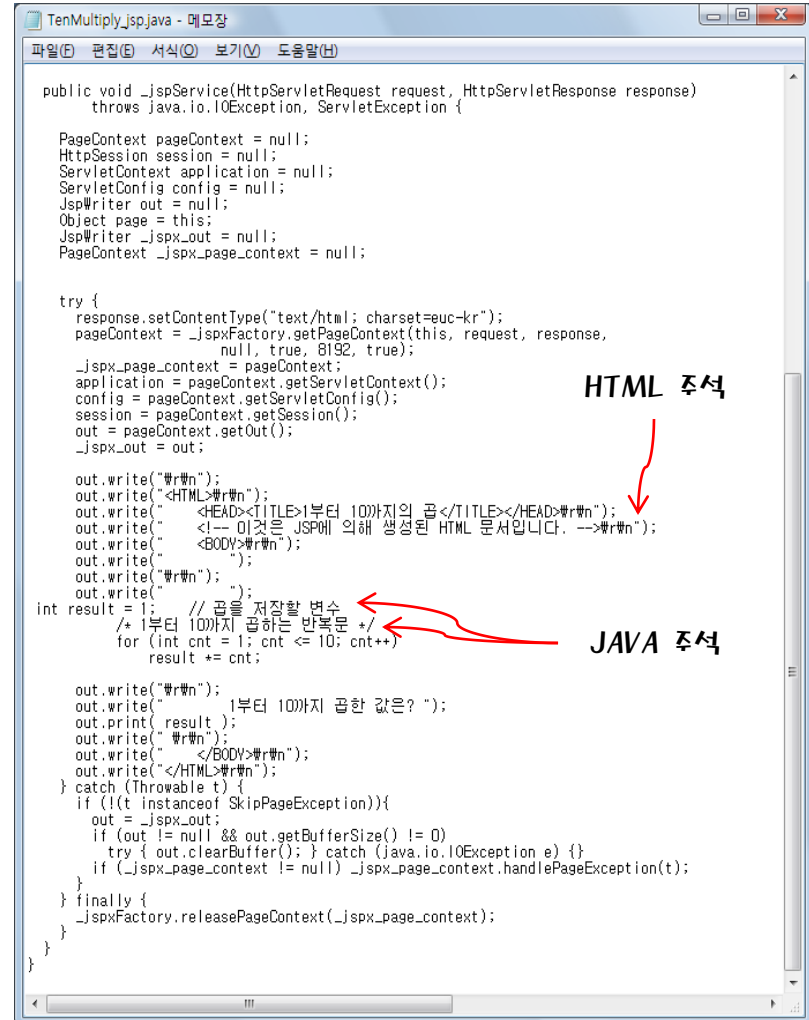
        result *= cnt;

    %>

    1부터 10까지 곱한 값은? <%= result %>

  </BODY>

</HTML>
```



```
TenMultiply.jsp.java - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

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

    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    PageContext _jspx_page_context = null;

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

        out.write("\r\n");
        out.write("<HTML>\r\n");
        out.write("<HEAD><TITLE>1부터 10까지의 곱</TITLE></HEAD>\r\n");
        out.write("<!-- 이것은 JSP에 의해 생성된 HTML 문서입니다. -->\r\n");
        out.write("<BODY>\r\n");
        out.write(" ");
        out.write("\r\n");
        out.write(" ");
        int result = 1; // 곱을 저장할 변수
        /* 1부터 10까지 곱하는 반복문 */
        for (int cnt = 1; cnt <= 10; cnt++)
            result *= cnt;

        out.write("\r\n");
        out.write(" 1부터 10까지 곱한 값은? ");
        out.print(result);
        out.write("\r\n");
        out.write("</BODY>\r\n");
        out.write("</HTML>\r\n");
    } catch (Throwable t) {
        if (!(t instanceof SkipPageException)){
            out = _jspx_out;
            if (out != null && out.getBufferSize() != 0)
                try { out.clearBuffer(); } catch (java.io.IOException e) {}
            if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
        }
    } finally {
        _jspxFactory.releasePageContext(_jspx_page_context);
    }
}
```

HTML 주석

JAVA 주석

- JSP 페이지의 내장 변수(implicit variable) : JSP 페이지 안에 선언을 하지 않고도 사용할 수 있는 변수

```
<%@page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>정수를 순서대로</TITLE></HEAD>
  <BODY>
    <H3>정수를 순서대로</H3>
    <%
      String str = request.getParameter( "MAX " );
      int max = Integer.parseInt(str);
      for (int cnt = 1; cnt <= max; cnt++)
        out.println(cnt + "<BR> ");
    %>
  </BODY>
</HTML>
```

내장 변수

내장 변수

[그림 3-15] JSP 페이지의 내장 변수의 예

- request 내장 변수는 서블릿 클래스의 doGet, doPost 메서드의 첫 번째 파라미터와 동일한 역할을 한다.
- out 내장 변수는 서블릿 클래스에서 getWriter 메서드를 호출해서 얻은 PrintWriter 객체와 마찬가지로의 역할을 한다.

- JSP 페이지 안에서 내장 변수를 사용할 수 있는 이유는 웹 컨테이너가 JSP 페이지를 서블릿 클래스로 변환할 때 자동으로 내장 변수를 선언하기 때문이다.
- JSP 페이지에서 사용할 수 있는 내장 변수들

변수 이름	제공하는 기능/변수의 역할	변수 타입
request	doGet, doPost 메서드의 첫 번째 파라미터와 동일한 역할	javax.servlet.http.HttpServletRequest
response	doGet, doPost 메서드의 두 번째 파라미터와 동일한 역할	javax.servlet.http.HttpServletResponse
out	웹 브라우저로 HTML 코드를 출력하는 기능	javax.servlet.jsp.JspWriter
application	JSP 페이지가 속하는 웹 애플리케이션에 관련된 기능	javax.servlet.ServletContext
config	JSP 페이지의 구성 정보를 가져오는 기능	javax.servlet.ServletConfig
pageContext	JSP 페이지 범위 내에서 사용할 수 있는 데이터 저장 기능 등	javax.servlet.jsp.PageContext
session	세션에 관련된 기능	javax.servlet.http.HttpSession
page	JSP 페이지로부터 생성된 서블릿	java.lang.Object
exception	익셉션 객체	java.lang.Throwable

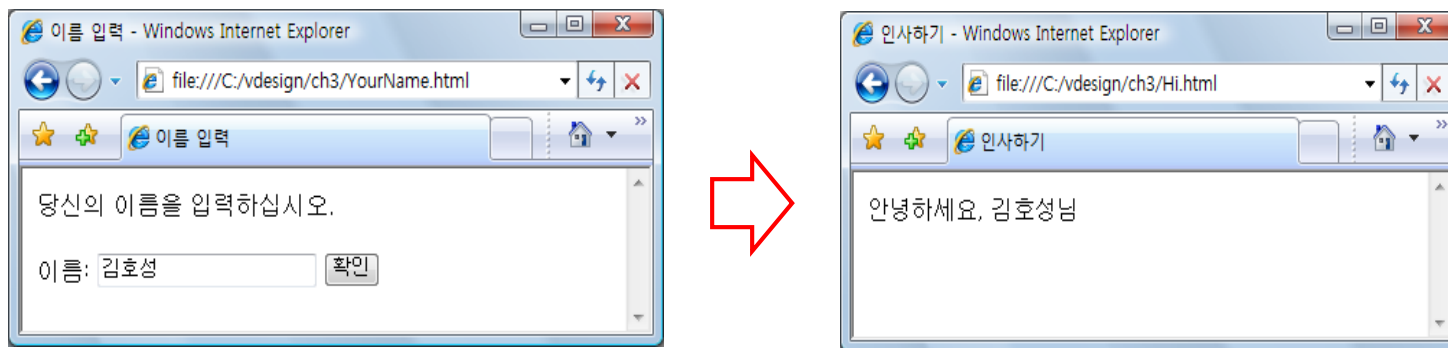
- request 내장 변수

- request 내장 변수는 서블릿 클래스에 있는 doGet, doPost 메서드의 첫 번째 파라미터와 동일한 역할을 하고, 타입도 동일하게 javax.servlet.http.HttpServletRequest이다.

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

↑
데이터 이름

- 웹 브라우저를 통해 입력된 데이터를 가져다가 처리하는 애플리케이션을 작성해보자.



[그림 3-17] 인사말을 출력하는 웹 애플리케이션의 화면 설계

- request 내장 변수

- 둘 이상의 화면으로 구성된 애플리케이션은 먼저 URL을 정한 뒤 각 URL에 해당하는 코드를 작성하는 것이 좋다.

http://localhost:8080/brain03/YourName.html

← (그림3-17) 왼쪽 화면 URL

http://localhost:8080/brain03/Hi.jsp

← (그림3-17) 오른쪽 화면 URL

YourName.html

[예제3-6] 웹 브라우저로부터 이름을 입력받는 HTML 문서

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type" content= "text/html; charset=euc-kr" >
    <TITLE>이름 입력</TITLE>
  </HEAD>
  <BODY>
    당신의 이름을 입력하세요.
    <FORM ACTION=/brain03/Hi.jsp METHOD=GET>
      이름: <INPUT TYPE=TEXT NAME=YOURNAME>
      <INPUT TYPE=SUBMIT VALUE= '확인' >
    </FORM>
  </BODY>
</HTML>
```

YourName.html

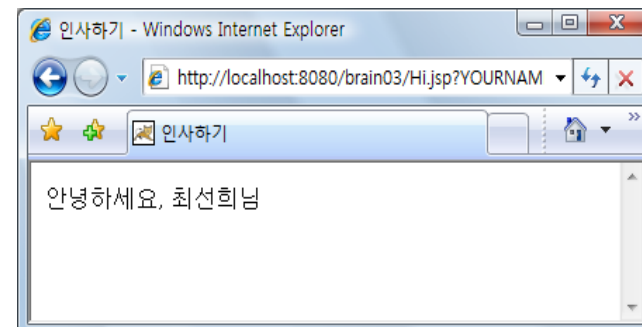
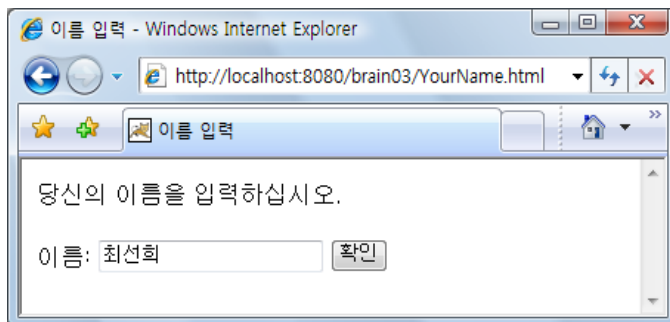
[예제3-6] 웹 브라우저로부터 이름을 입력받는 HTML 문서

```
<HTML>
  <HEAD>
    <META http-equiv="Content-Type" content=
"text/html; charset=euc-kr">
    <TITLE>이름 입력</TITLE>
  </HEAD>
  <BODY>
    당신의 이름을 입력하세요.
    <FORM ACTION=hi.jsp METHOD=GET>
      이름: <INPUT TYPE=TEXT NAME=YOURNAME>
      <INPUT TYPE=SUBMIT VALUE='확인'>
    </FORM>
  </BODY>
</HTML>
```

hi.jsp

[예제3-7] 입력된 이름을 가지고 인사말을 출력하는 JSP 페이지

```
<% @page contentType="text/html; charset=euc-kr"%>
<HTML>
  <HEAD><TITLE>인사하기</TITLE></HEAD>
  <BODY>
    안녕하세요, <%= request.getParameter("YOURNAME") %>님
  </BODY>
</HTML>
```



[그림 3-18] 예제 3-6, 예제 3-7의 실행 결과

- login_form.html 을 통해 전달되 id, pw를 login_form_result.jsp 에서 출력하세요

login_form.html

아이디

비밀번호

login_form_result.jsp

id:apple
password:orange

```
<body>
<form method="post" action=" " >
  <ul>
    <li>아이디 </li>
    <li><input type="text" name="id"></input></li>
    <li>비밀번호</li>
    <li><input type="text" name="passwd"></input></li>
    <li><input type="submit" value="로그인"></input></li>
  </ul>
</form>
</body>
```

- **response 내장 변수**

- response 내장 변수는 서블릿 클래스에 있는 doGet, doPost 메서드의 두 번째 파라미터와 동일한 역할을 한다. 이 변수는 javax.servlet.http.HttpServletResponse 타입이기 때문에 이 인터페이스에 속하는 여러 가지 메서드들을 호출 할 수 있다.

```
response.sendRedirect( "http://www.hanb.co.kr/ ");
```

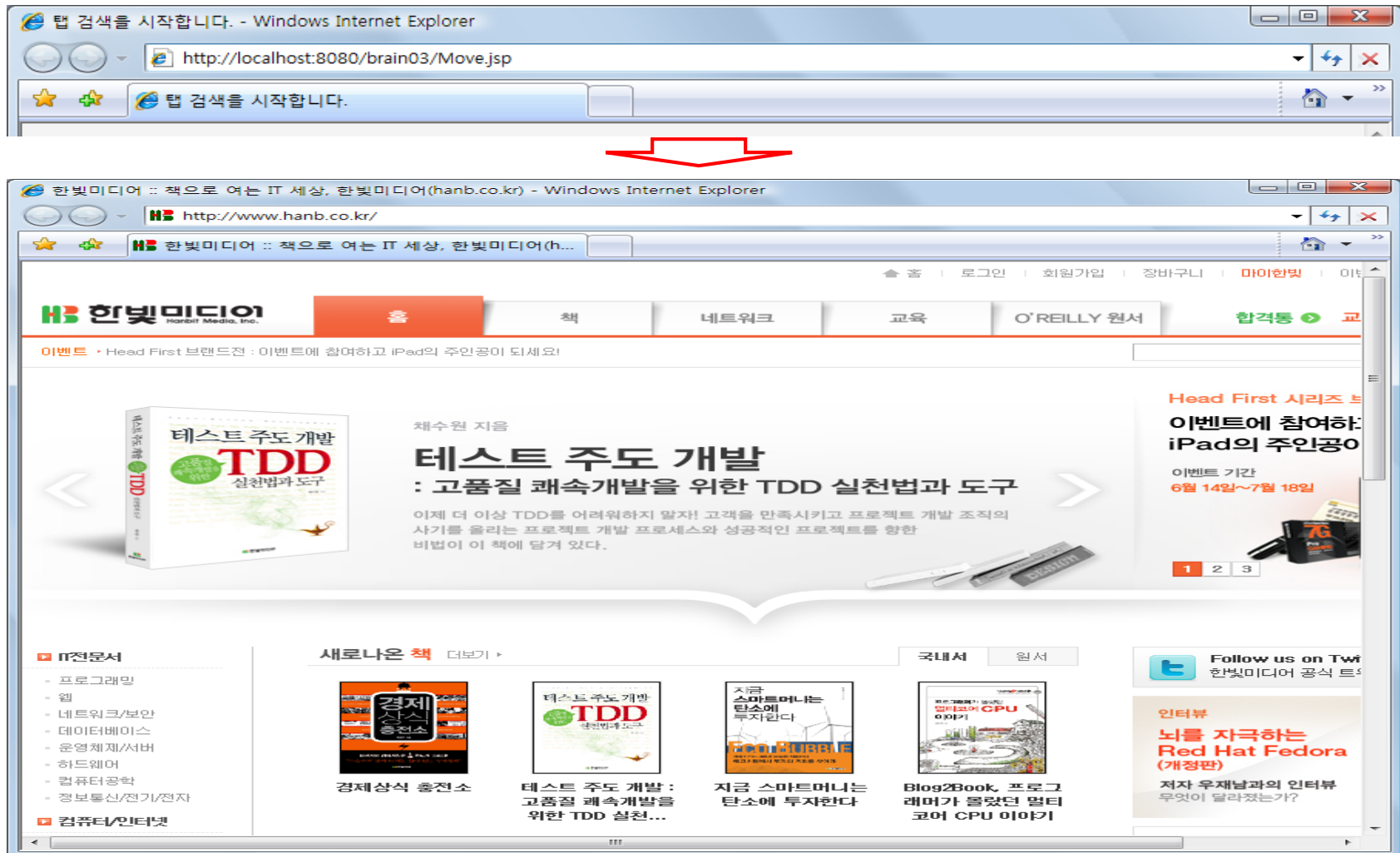
호출할 웹 자원의 URL

- sendRedirect 메서드를 호출할 때 주의할 점: 이 메서드를 호출하기 전과 후에 웹 브라우저로 데이터를 출력하면 안 된다.

[예제3-11] 다른 웹 페이지로 이동하는 JSP 페이지

```
<%@page contentType="text/html; charset=euc-kr"%>
<HTML>
  <HEAD><TITLE>response 내장변수</TITLE></HEAD>
  <BODY>
    <% response.sendRedirect( "http://www.hanb.co.kr/ "); %>
  </BODY>
</HTML>
```

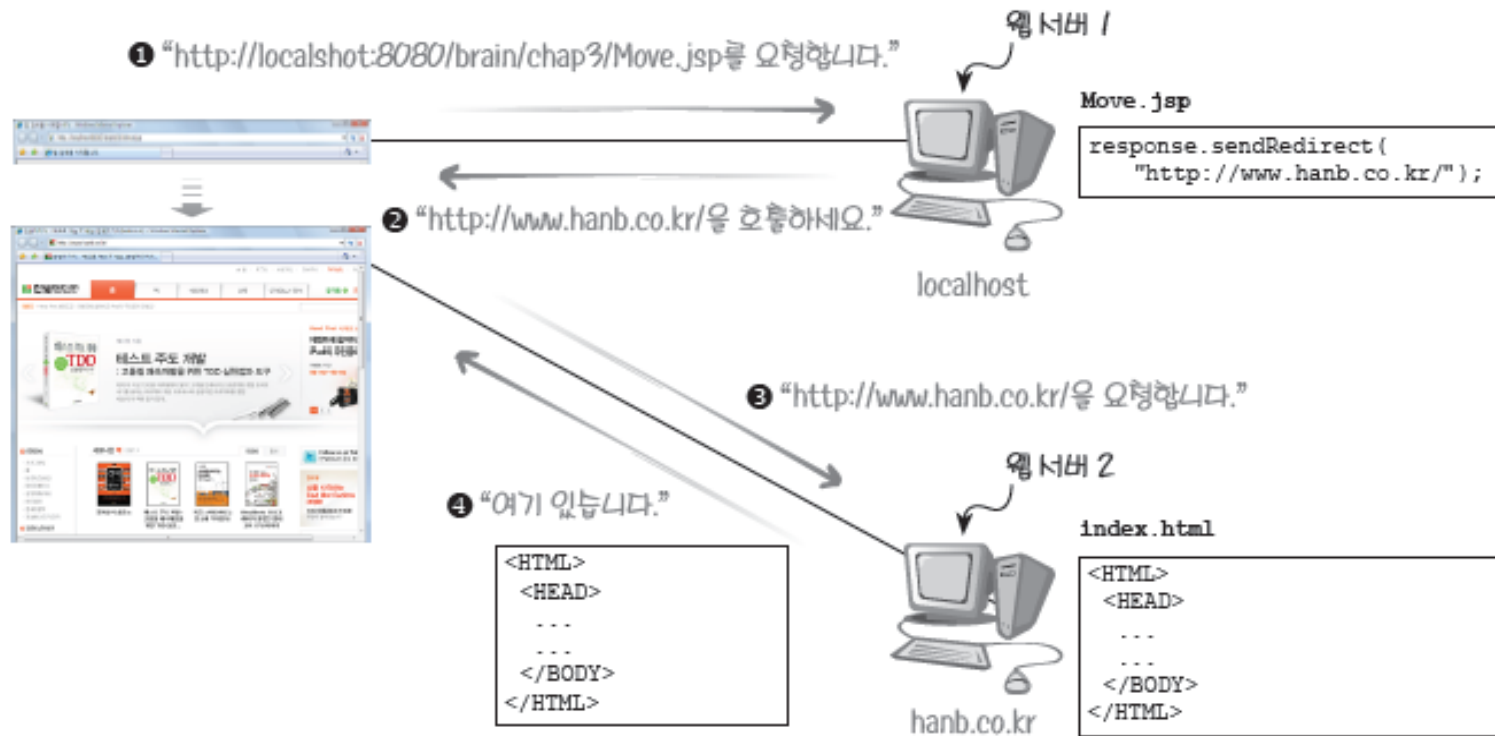
- response 내장 변수



[그림 3-22] 예제 3-11의 실행 결과

- response 내장 변수

- sendRedirect 메서드는 파라미터로 지정한 URL을 직접 호출하는 것이 아니라 그 URL을 이용해서 다시 웹 자원을 호출하라는 메시지를 웹 브라우저로 보낼 뿐이다.



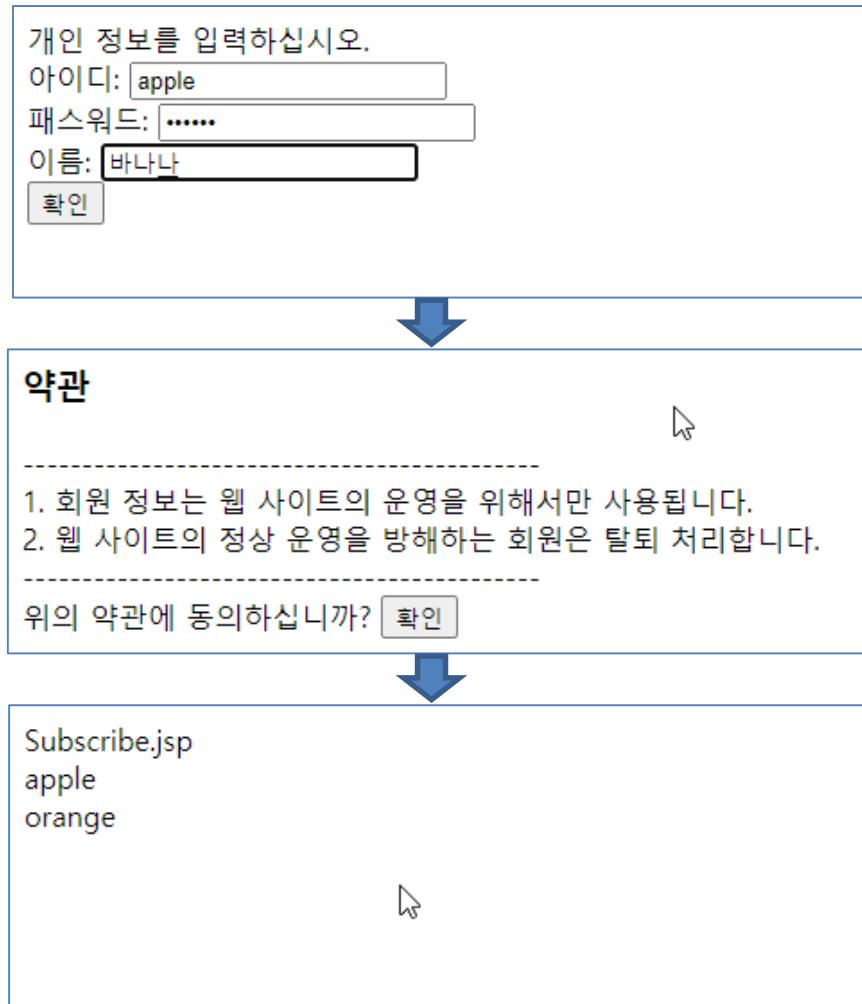
[그림 3-23] sendRedirect 메서드의 작동 원리

- **session 내장객체**

- HTTP 프로토콜이 비연결형 프로토콜이기 때문에 한 페이지가 출력된 다음에는 클라이언트와 서버의 연결이 끊어진다. 따라서 한번 로그인한 사용자가 로그아웃할 때까지 페이지를 이동해도 보관해야 할 정보가 있다면 이에 대한 처리가 매우 곤란해진다.
- 이러한 HTTP 프로토콜 문제점을 해결하려고 나온 것이 쿠키와 세션이다.
- session 은 javax.servlet.http.HttpSession 인터페이스의 참조 변수 이다.
- session 은 접속하는 사용자 별로 따로 생성되며 일정시간 유지되고 소멸된다.
- 이러한 세션의 특징을 이용해 setAttribute() 메서드를 이용해 임의의 값을 저장해 놓고 활용할 수 있음.
- 세션이 주로 사용되는 경우는 다음과 같다.
 - ① 사용자 로그인 후 세션을 설정하고 일정 시간이 지난 경우 다시 사용자 인증을 요구 할 때.
 - ② 쇼핑몰에서 장바구니 기능을 구현할 때.
 - ③ 사용자의 페이지 이동 동선 등 웹 페이지 트래킹 분석 기능 등을 구현할 때.

■ session 주요 메소드

메서드	설명
<code>getId()</code>	각 접속에 대한 세션 고유의 ID를 문자열 형태로 반환한다.
<code>getCreatingTime()</code>	세션 생성 시간을 January 1, 1970 GMT.부터 long형 밀리세컨드 값으로 반환한다.
<code>getLastAccessedTime()</code>	현재 세션으로 마지막 작업한 시간을 long형 밀리세컨드 값으로 반환한다.
<code>getMaxInactiveInterval()</code>	세션의 유지 시간을 초로 반환한다. 이를 통해 세션의 유효 시간을 알 수 있다.
<code>setMaxInactiveInterval(t)</code>	세션의 유효 시간을 t에 설정된 초 값으로 설정한다.
<code>invalidate()</code>	현재 세션을 종료한다. 세션과 관련된 값들은 모두 지워진다.
<code>getAttribute(attr)</code>	문자열 attr로 설정된 세션 값을 <code>java.lang.Object</code> 형태로 반환한다.
<code>setAttribute(name,attr)</code>	문자열 name으로 <code>java.lang.Object</code> attr을 설정한다.



[그림 4-20] 회원 가입 애플리케이션의 화면 설계

- 서블릿 클래스에서는 새로운 세션을 시작하거나 진행 중인 세션을 계속하기 위해서는 `getSession` 메서드를 호출해야 하지만, JSP 페이지에서는 JSP 페이지가 서블릿 클래스로 변환되는 과정에서 이 메서드를 호출하는 코드가 자동으로 추가 되기 때문에 `getSession` 메서드를 호출 할 필요가 없다.
- `session` 내장 변수를 사용하면 세션 데이터 영역에 데이터를 저장할 수도 있고, 그 영역에 있는 데이터를 읽어오거나 삭제할 수도 있다.

```
session.setAttribute( "ID ", "lee77 ");
```

세션 데이터를 저장하는 메서드

```
String str = (String) session.getAttribute( "ID " );
```

세션 데이터를 가져오는 메서드

```
session.removeAttribute( "ID " );
```

세션 데이터를 삭제하는 메서드

- 세션을 끝내려면 session 내장 변수에 대해 invalidate라는 메서드를 호출하면 된다.

```
session.invalidate();
```

세션을 끝내는 메서드

http://localhost:8080/jsp_lecture/ex8_session/PersonalInfo.html

(그림 4-20)의 첫 번째 화면의 URL

http://localhost:8080/jsp_lecture/ex8_session/Agreement.jsp

(그림 4-20)의 두 번째 화면의 URL

http://localhost:8080/jsp_lecture/ex8_session/Subscribe.jsp

회원 정보를 출력하는 JSP 페이지의 URL

- 앞 페이지의 네 URL에 해당하는 HTML 문서와 JSP 페이지는 다음과 같이 작성하면 된다.

PersonalInfo.html

[예제4-12] 개인 정보 입력 화면을 제공하는 HTML 문서

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type " content= "text/html; charset=euc-kr ">
    <TITLE>회원 가입</TITLE>
  </HEAD>
  <BODY>
    개인 정보를 입력하세요.
    <FORM ACTION=Agreement.jsp METHOD=POST>
      아이디: <INPUT TYPE=TEXT NAME=ID><BR>
      패스워드: <INPUT TYPE=PASSWORD NAME=PASSWORD><BR>
      이름: <INPUT TYPE=TEXT NAME=NAME><BR>
      <INPUT TYPE=SUBMIT VALUE= '확인' >
    </FORM>
  </BODY>
</HTML>
```


Agreement.jsp

[예제4-13] 약관 동의 화면을 제공하는 JSP 페이지

```
<% @page contentType="text/html; charset=euc-kr"%>
<%
    request.setCharacterEncoding("euc-kr");
    String id = request.getParameter("ID");
    String password = request.getParameter("PASSWORD");
    String name = request.getParameter("NAME");
    session.setAttribute("ID", id);
    session.setAttribute("PASSWORD", password);
    session.setAttribute("NAME", name);
%>
<HTML>
<HEAD><TITLE>회원 가입</TITLE></HEAD>
<BODY>
<H3>약관</H3>
----- <BR>
1. 회원 정보는 웹 사이트의 운영을 위해서만 사용됩니다. <BR>
2. 웹 사이트의 정상 운영을 방해하는 회원은 탈퇴 처리합니다. <BR>
----- <BR>
<FORM ACTION=Subscribe.jsp METHOD=POST>
    위의 약관에 동의하십니까?
    <INPUT TYPE=SUBMIT VALUE='확인'>
</FORM>
</BODY>
</HTML>
```

세션 데이터를 저장합니다

Subscribe.jsp

[예제4-14] 회원 정보를 저장하는 JSP 페이지

```
<% @page contentType="text/html; charset=euc-kr"%>
<% @page import="java.io.*"%>
<%
    String result="success";
    String id = (String) session.getAttribute("ID");
    String password = (String) session.getAttribute("PASSWORD");
    String name = (String) session.getAttribute("NAME");
    session.invalidate();
%>

Subscribe.jsp
<%= id %><br>
<%= password %><br>
<%= name %><br>
```

아이디

다음

id를 getParameter로 받아서
세션에 저장

패스워드

다음

pw를 getParameter로 받아서
세션에 저장

동의 하니까?

다음

세션에서 id, pw를 얻어서 출력

가입 완료
apple
orange

- **application 내장 변수**

- application 내장 변수는 웹 애플리케이션에 관련된 여러 가지 기능을 제공한다.
- application 내장 변수에 대해 호출할 수 있는 `getContextPath` 메서드는 웹 애플리케이션의 URL 경로명을 리턴하는 메서드이다.

```
String appPath = application.getContextPath();
```

↑
웹 애플리케이션의 URL 경로명을 리턴하는 메서드

- application 내장 변수에 대해 호출할 수 있는 `getRealPath` 메서드는 웹 애플리케이션 내에서의 파일 경로명을 파일시스템 전체에 대한 절대 경로명으로 바꾸는 메서드이다.

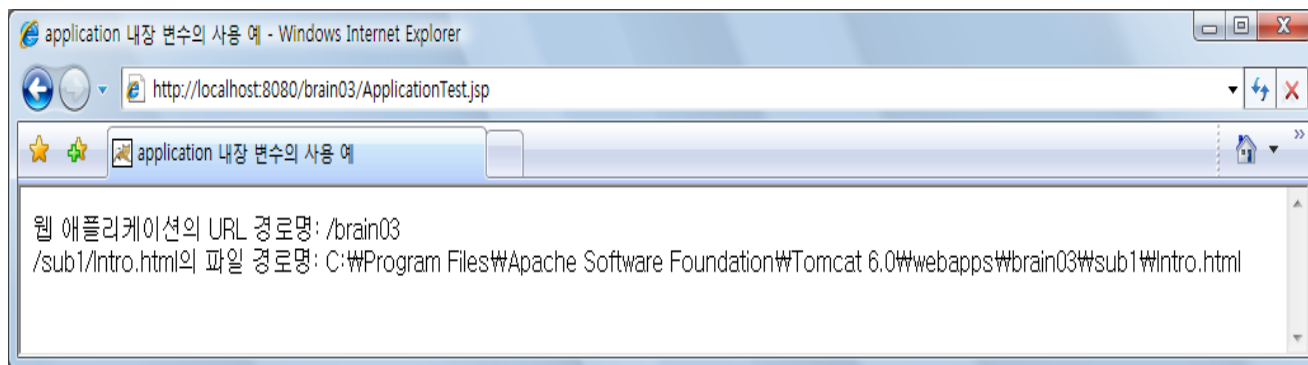
```
String absolutePath = application.getRealPath( "/sub1/Intro.html ");
```

↑
웹 애플리케이션 내에서의 파일의 경로명

- Application 내장 변수

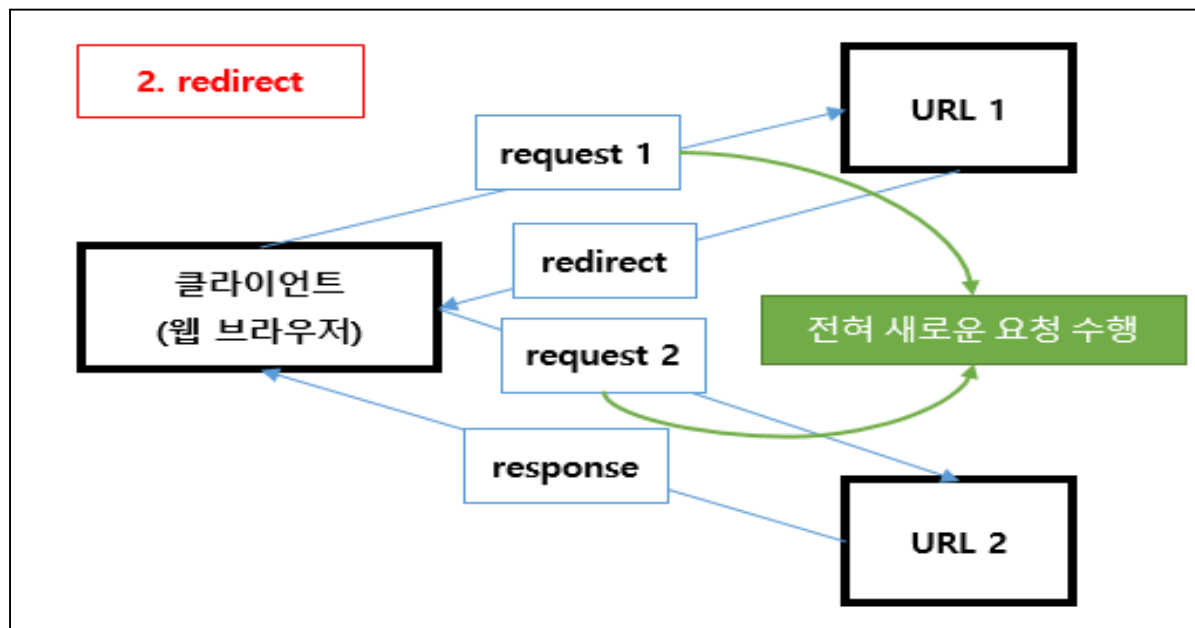
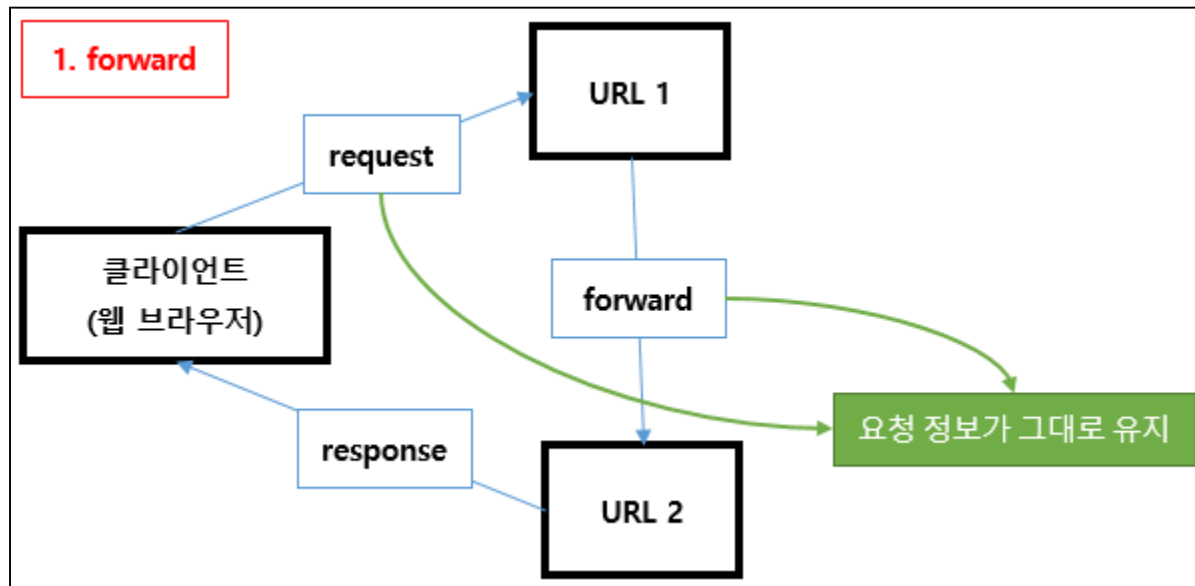
[예제3-12] application 내장 변수의 사용 예

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>application 내장 변수의 사용 예</TITLE></HEAD>
  <BODY>
    <%
      String appPath = application.getContextPath();
      String filePath = application.getRealPath( "/sub1/Intro.html " );
    %>
    웹 애플리케이션의 URL 경로명: <%= appPath %> <BR>
    /sub1/Intro.html의 파일 경로명: <%= filePath %> <BR>
  </BODY>
</HTML>
```



[그림 3-24] 예제 3-12의 실행 결과

다른 JSP 페이지 호출하기



- **forward 메서드의 사용 방법**

- forward 메서드는 JSP 페이지 안에서 다른 JSP 페이지를 호출할 때 사용하는 메서드이다.
- 이 메서드는 `javax.servlet.RequestDispatcher` 인터페이스에 속하기 때문에 이 타입의 객체가 있어야 호출할 수 있다.

```
RequestDispatcher dispatcher = request.getRequestDispatcher( "Result.jsp ");
```

↑
호출할 JSP 페이지의 URL 경로명

- forward 메서드를 호출할 때에는 request 내장 변수와 response 내장 변수를 파라미터로 넘겨줘야 한다.

```
dispatcher.forward(request, response);
```

request 내장 변수

response 내장 변수

- **forward 메서드의 사용 방법**

- forward 메서드를 통해 호출하는 JSP 페이지로 데이터를 넘겨주려면 이 메서드보다 먼저 request.setAttribute 메서드를 호출해서 request 내장 변수 안에 데이터를 저장해 놓아야 한다.

```
request.setAttribute( "HEIGHT ", new Integer(178));
```

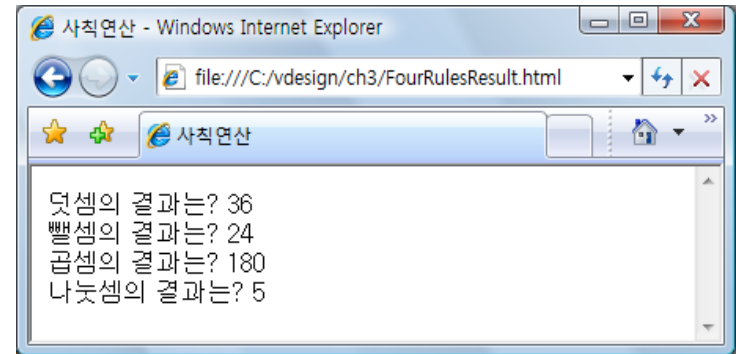
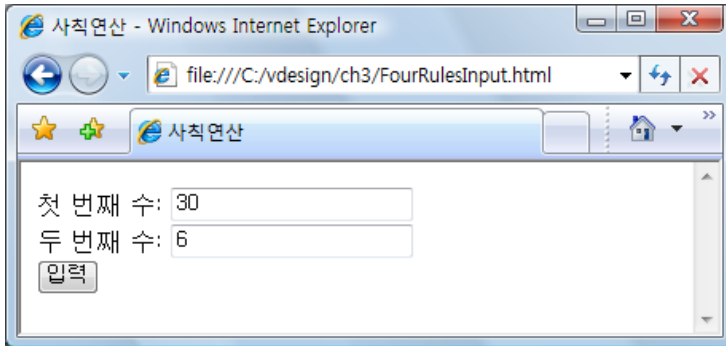
데이터 이름 데이터 값

- 호출된 JSP 페이지 안에서 request 내장 변수 안의 데이터를 가져오려면 request.getAttribute 메서드를 호출하면 된다.

```
Integer height = (Integer) request.getAttribute( "HEIGHT " );
```

캐스트 연산자 데이터 이름

- forward 메서드의 사용 방법



[그림 3-32] 사칙 연산을 수행하는 웹 애플리케이션 화면 설계

(그림 3-32)의 왼쪽 화면의 URL

<http://localhost:8080/brain03/FourRulesInput.html>



(그림 3-32)의 오른쪽 화면의 URL

<http://localhost:8080/brain03/FourRules.jsp>



사칙 연산을 수행하는 JSP 페이지의 URL

<http://localhost:8080/brain03/FourRuelsResult.jsp>

사칙 연산의 결과를 출력하는 JSP 페이지의 URL

- **forward 메서드의 사용 방법**

FourRulesInput.html

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type " content= "text/html; charset=euc-kr ">
    <TITLE>사칙 연산</TITLE>
  </HEAD>
  <BODY>
    <FORM ACTION=FourRules.jsp>
      첫째 수: <INPUT TYPE=TEXT NAME=NUM1><BR>
      둘째 수: <INPUT TYPE=TEXT NAME=NUM2><BR>
      <INPUT TYPE=SUBMIT VALUE= '입력' >
    </FORM>
  </BODY>
</HTML>
```

FourRules.jsp

```
<%  
    String str1 = request.getParameter("NUM1");  
    String str2 = request.getParameter("NUM2");  
    int num1 = Integer.parseInt(str1);  
    int num2 = Integer.parseInt(str2);  
    request.setAttribute("SUM", new Integer(num1 + num2));  
    request.setAttribute("DIFFERENCE", new Integer(num1 - num2));  
    request.setAttribute("PRODUCT", new Integer(num1 * num2));  
    request.setAttribute("QUOTIENT", new Integer(num1 / num2));  
    RequestDispatcher dispatcher = request.getRequestDispatcher("FourRulesResult.jsp");  
    dispatcher.forward(request, response);  
%>
```

FourRulesResult.jsp

```
<% @page contentType="text/html; charset=euc-kr"%>
<HTML>
  <HEAD><TITLE>사칙연산</TITLE></HEAD>
  <BODY>
    덧셈의 결과는? <%= request.getAttribute("SUM") %> <BR>
    뺄셈의 결과는? <%= request.getAttribute("DIFFERENCE") %> <BR>
    곱셈의 결과는? <%= request.getAttribute("PRODUCT") %> <BR>
    나눗셈의 결과는? <%= request.getAttribute("QUOTIENT") %> <BR>
  </BODY>
</HTML>
```

아이디

apple

비밀번호

orange

로그인

login_form.jsp



login_forward.jsp id, pw를 세션에 저장후 result.jsp 로 forwarding



result.jsp

apple

orange

login_result.jsp

- **sendRedirect 메서드의 사용 방법**

FourRulesInput.html

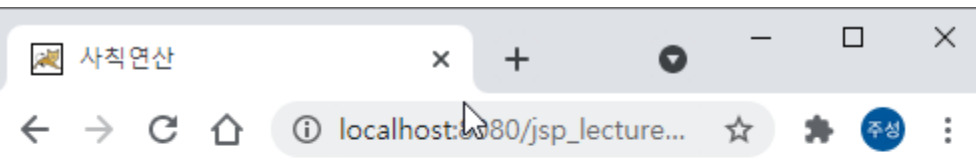
```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type " content= "text/html; charset=euc-kr ">
    <TITLE>사칙 연산</TITLE>
  </HEAD>
  <BODY>
    <FORM ACTION=FourRules.jsp>
      첫 번째 수: <INPUT TYPE=TEXT NAME=NUM1><BR>
      두 번째 수: <INPUT TYPE=TEXT NAME=NUM2><BR>
      <INPUT TYPE=SUBMIT VALUE= '입력' >
    </FORM>
  </BODY>
</HTML>
```

FourRules.jsp

```
<%  
    String str1 = request.getParameter("NUM1");  
    String str2 = request.getParameter("NUM2");  
    int num1 = Integer.parseInt(str1);  
    int num2 = Integer.parseInt(str2);  
    request.setAttribute("SUM", new Integer(num1 + num2));  
    request.setAttribute("DIFFERENCE", new Integer(num1 - num2));  
    request.setAttribute("PRODUCT", new Integer(num1 * num2));  
    request.setAttribute("QUOTIENT", new Integer(num1 / num2));  
    response.sendRedirect("FourRulesResult.jsp");  
%>
```

FourRulesResult.jsp

```
<% @page contentType="text/html; charset=euc-kr"%>
<HTML>
  <HEAD><TITLE>사칙연산</TITLE></HEAD>
  <BODY>
    덧셈의 결과는? <%= request.getAttribute("SUM") %> <BR>
    뺄셈의 결과는? <%= request.getAttribute("DIFFERENCE") %> <BR>
    곱셈의 결과는? <%= request.getAttribute("PRODUCT") %> <BR>
    나눗셈의 결과는? <%= request.getAttribute("QUOTIENT") %> <BR>
  </BODY>
</HTML>
```



덧셈의 결과는? null
뺄셈의 결과는? null
곱셈의 결과는? null
나눗셈의 결과는? null