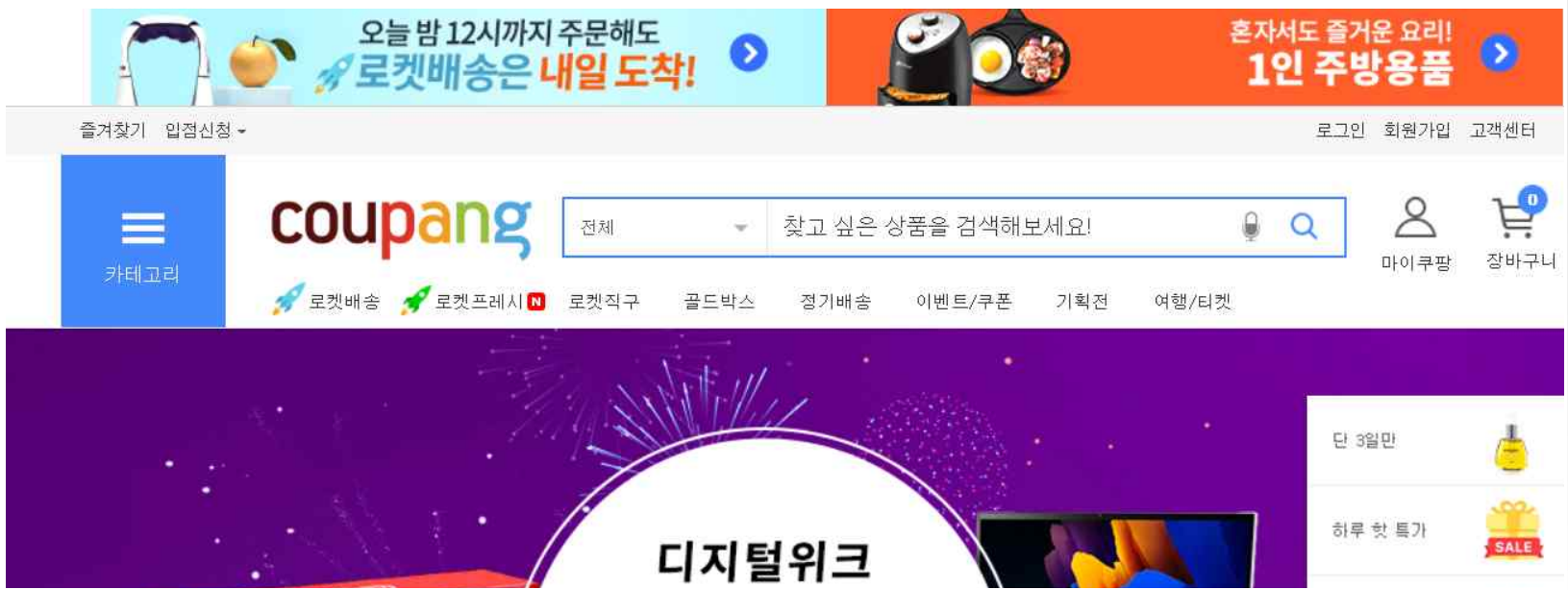
제5장
JSP- Ⅰ

1. JSP 등장 배경

1. 서블릿으로 화면 구현 시 문제점

- 기존 서블릿에서는 자바 코드를 기반으로 문자열을 사용해 HTML과 자바스크립트로 화면을 구현했음
- JSP는 이와 반대로 HTML, CSS와 자바스크립트를 기반으로 JSP 요소들을 사용해 화면을 구현함



1. JSP 등장 배경

기본 웹 사이트의 제작 소스

```
19
20      <script language="Javascript">
21      if (document.location.protocol == 'http:') {
22          document.location.href = document.location.href.replace('http:', 'https:');
23      }
24      </script>
25
26
27
28      <style>
29
30
31      html, body{
32          max-width: 100%;
33          overflow-x: hidden;
34          width:100%;
35          height:100%;
36          margin:0;
37          padding:0;
38      }
39      </style>
40
41      <script src="/js/h.js" type="text/javascript"></script>
42
43  </head>
44
45
46  <body scroll="no" style="overflow: hidden; -ms-overflow-y: hidden;" onload="wph.init();" onresize="/*wph.init();*/">
47      <iframe name="mainFrame" width="100%" height="100%" id="mainFrame" src="/html/main/main.asp" border="0" scrolling="yes"
    allowfullscreen="allowfullscreen" style="-ms-zoom: 1;" webkitallowfullscreen="webkitallowfullscreen"
```

1. JSP 등장 배경

문제점

- 웹 프로그램의 화면 기능이 복잡해지므로 서블릿의 자바 기반으로 화면 기능 구현 시 어려움이 발생함
- 디자이너 입장에서 화면 구현 시 자바 코드로 인해 작업이 어려워함
- 서블릿에 비즈니스 로직과 화면 기능이 같이 있다 보니 개발 후 유지관리가 불편함

해결책

- 서블릿의 비즈니스 로직과 결과를 보여주는 화면 기능을 분리하자!
- 비즈니스 로직과 화면을 분리함으로써 개발자는 비즈니스 로직 구현에 집중하고, 디자이너는 화면 기능 구현에만 집중하자!
- 개발 후 재사용성과 유지관리가 훨씬 수월해진다!

2. JSP의 구성 요소

2. JSP의 구성 요소

- HTML 태그, CSS 그리고 자바스크립트 코드
- JSP 기본 태그
- JSP 액션 태그
- 개발자가 직접 만들거나 프레임워크에서 제공하는 커스텀(custom) 태그

2-1. JSP태그의 개념 이해

Servlet은 JAVA언어를 이용하여 문서를 작성하고, 출력 객체(PrintWriter 등)를 이용하여 HTML코드를 삽입했다.
JSP는 Servlet과 반대로 HTML코드에 JAVA언어를 삽입하여 동적 문서를 만들 수 있다.
HTML코드 안에 JAVA코드를 삽입하기 위해서는 태그를 이용해야 하며, 이러한 태그를 공부해야 한다.

JSP태그 종류

지시자 : <%@ %> : 페이지 속성(ex. import)
주석 : <%-- --%> : HTML주석 소스보기 하면 보인다.
반면에 JSP주석은 보이질 않는다.(서버 단)
선언 : <%! %> : 변수, 메서드 선언(전역 성질을 지님)
표현식 : <%= %> : 결과값 출력
스크립트릿 : <% %> : JAVA 코드(가장 많이 사용된다)
액션태그 : <jsp:action> </jsp:action> : 자바빈(클래스) 연결

HTML파일 내에 JSP코드를 삽입하여 동적으로 작동하게 한다.(JSP파일)

맨 첫 강의 때 MVC패턴에 대해서 언급한 적이 있다.
보통 JSP는 View단을 위해 존재한다고 보고, Servlet은 Controller역할을 한다고 했다.
왜 JSP가 View단에 적용이 쉬운가? 바로 HTML태그를 이용해 화면 구성이 쉽기 때문인 것이고,
아울러 서블릿은 자바코드로 되어 있어 로직 수행코드가 들어감으로써 Controller역할을 하는 것이다.

2-2. JSP의 3단계 작업 과정

톰캣 컨테이너에서 JSP 변환 과정

1. 변환 단계(Translation Step): 컨테이너는 JSP 파일을 자바 파일로 변환
2. 컴파일 단계(Compile Step): 컨테이너는 변환된 자바(java) 파일을 클래스(class) 파일로 컴파일
3. 실행 단계(Interpret Step): 컨테이너는 class 파일을 실행하여 그 결과(HTML, CSS와 자바스크립트 코드)를 브라우저로 전송해 출력

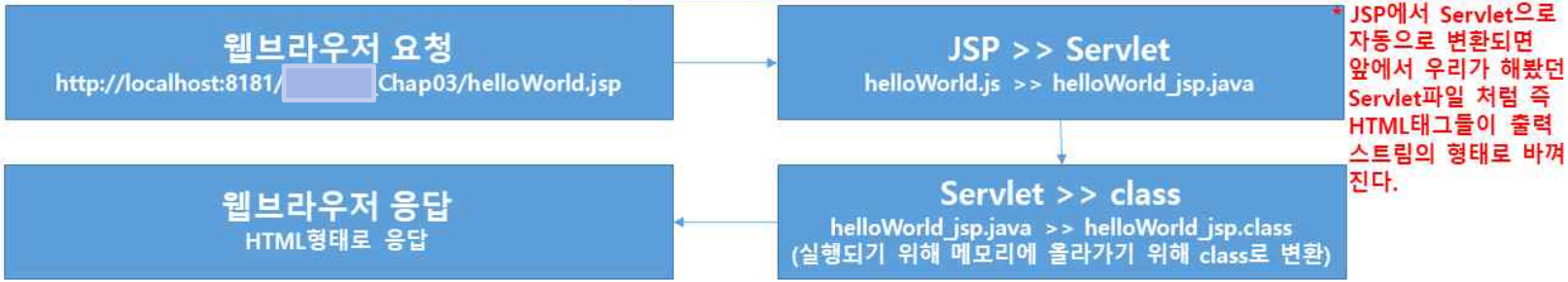
2-2. JSP의 3단계 작업 과정

JSP가 요청되어 응답하기까지의 과정을 이해하면, 개발에 많은 도움이 된다.

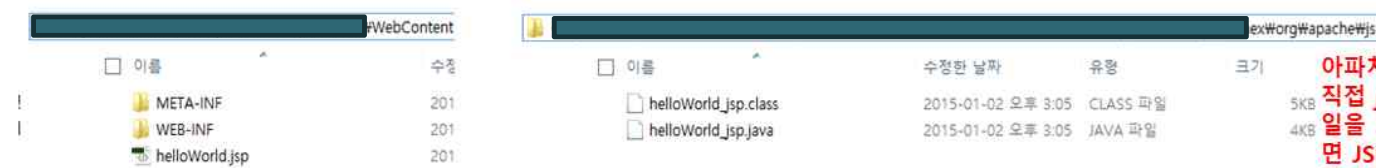
중요함) 클라이언트가 웹브라우저로 helloWorld.jsp를 요청하게 되면 WAS의 JSP컨테이너가 JSP파일을 Servlet파일(.java)로 자동 변환하게 한다.

그리고 변환된 Servlet파일(.java)은 컴파일 된 후 클래스파일(.class)로 변환되고, 요청한 클라이언트한테 html파일 형태로 응답된다.

기억할 것은 앞에서 서블릿 파일(자바코드)은 스레드를 생성해서 클라이언트에게 응답을 해준다고 했다. 재 요청이 들어와도 역시 그 스레드를 이용하는 것이다. JSP파일 역시 맨 첨 한 번만 메모리에 로딩되기 위해 서블릿으로 변환되어 클라이언트에게 응답을 해준다. 또한 재 요청이 들어와도 생성된 클래스로 응답을 한다.그래서 처음만 서블릿과 JSP의 속도가 약간 나지만 이후로는 거의 동일하다고 보면 된다.



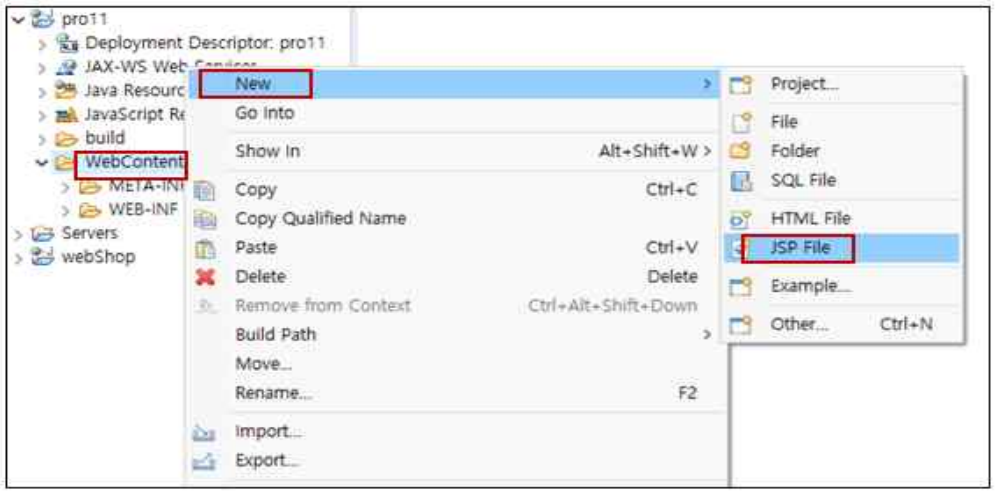
JSP에서 Servlet으로 자동으로 변환되면 앞에서 우리가 해봤던 Servlet파일 처럼 즉 HTML태그들이 출력 스트림의 형태로 바껴진다.



아파치 서버폴더에 가서 직접 java로 변환된 파일을 노트패드로 열어보면 JSP가 자바코드로 바뀐것을 확인할 수 있다.

2-2. JSP의 3단계 작업 과정

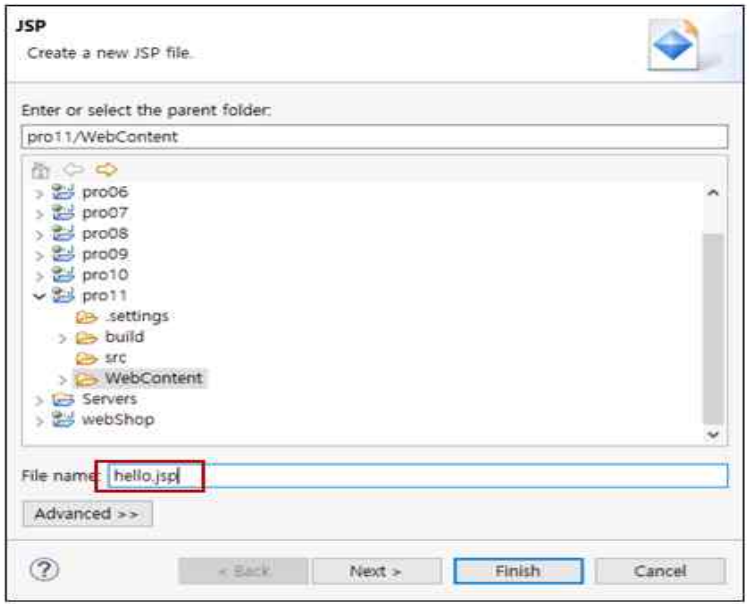
1. 이클립스에서 새 프로젝트를 만들고 WebContent 폴더에서 마우스 오른쪽 버튼을 클릭한 후 New > JSP File을 선택합니다.



❖ 반드시 servlet_api.jar을 설정해 줍니다

2-2. JSP의 3단계 작업 과정

2. 파일 이름으로 hello.jsp를 입력한 후 Finish를 클릭합니다.



2-2. JSP의 3단계 작업 과정

3. 생성된 JSP 파일에 간단한 HTML 태그와 메시지를 작성합니다.

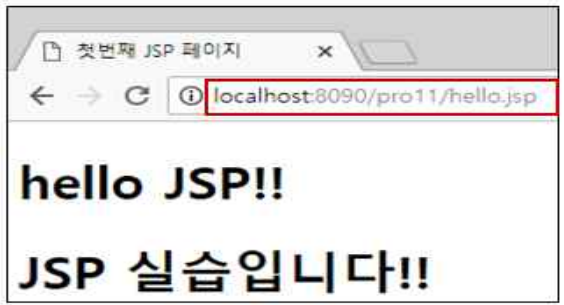
코드 11-2 pro11/WebContent/hello.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>첫 번째 JSP 페이지</title>
</head>
<body>
  <h1>hello JSP!!</h1>
  <h1>JSP 실습입니다!!</h1>
</body>
</html>
```

2-2. JSP의 3단계 작업 과정

4. 톰캣 컨테이너에 프로젝트를 추가합니다. 톰캣을 실행한 후 브라우저에서 HTML 파일을 요청 하듯이 JSP파일을 요청합니다.

- http://ip주소:포트번호/프로젝트이름/JSP파일이름

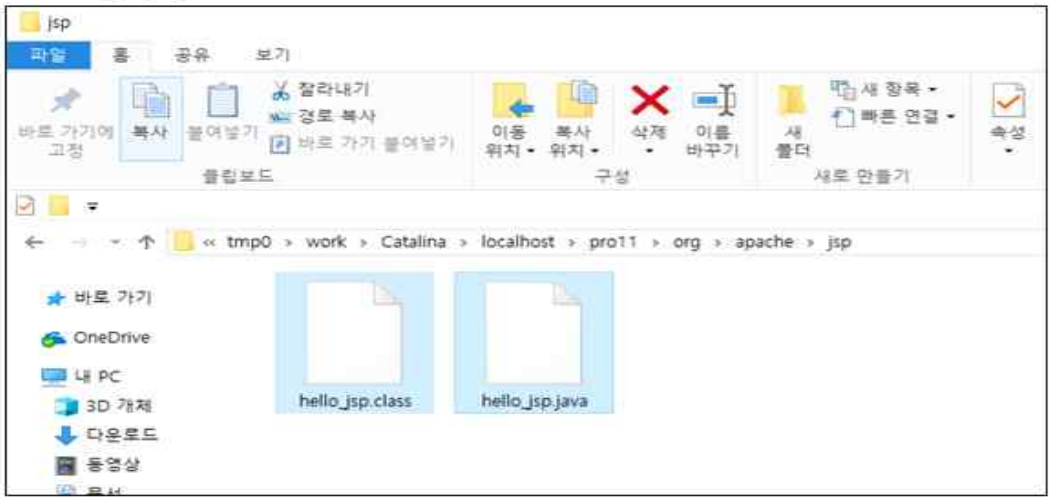


2-2. JSP의 3단계 작업 과정

hello.jsp 출력 과정

2-2. JSP의 3단계 작업 과정

hello_jsp.java 파일로 변환된 상태



❖ 이클립스에서 실행 시 자바 파일과 클래스 파일 생성 위치

```
%이클립스_workspace%\.metadata\.plugins\
org.eclipse.wst.server.core\tmp0 \work\Catalina\localhost\Practice_Chap05\org\apache\jsp
```

2-2. JSP의 3단계 작업 과정

hello_jsp.java로 변환한 후 브라우저로 전송한 HTML 태그

```
105     try {
106       response.setContentType("text/html; charset=UTF-8");
107       pageContext = _jspxFactory.getPageContext(this, request, response,
108             null, true, 8192, true);
109       _jspx_page_context = pageContext;
110       application = pageContext.getServletContext();
111       config = pageContext.getServletConfig();
112       session = pageContext.getSession();
113       out = pageContext.getOut();
114       _jspx_out = out;
115
116       out.write("\r\n");
117       out.write("<!DOCTYPE html>\r\n");
118       out.write("<html>\r\n");
119       out.write("<head>\r\n");
120       out.write("<meta charset=\"UTF-8\">\r\n");
121       out.write("<title>첫번째 JSP 페이지</title>\r\n");
122       out.write("</head>\r\n");
123       out.write("<body>\r\n");
124       out.write("  <h1>hello JSP!!</h1>\r\n");
125       out.write("  <h1>JSP 실습입니다!!</h1>\r\n");
126       out.write("</body>\r\n");
127       out.write("</html>\r\n");
128       out.write("\r\n");
129       out.write("\r\n");
130       out.write("\r\n");
131       out.write("\r\n");
132       out.write("\r\n");
133       out.write("\r\n");
134     } catch (java.lang.Throwable t) {
```

브라우저로 전송된 HTML 태그

view-source:localhost:8090/pro11/hello.jsp

```
1
2  <!DOCTYPE html>
3  <html>
4  <head>
5  <meta charset="UTF-8">
6  <title>첫번째 JSP 페이지</title>
7  </head>
8  <body>
9    <h1>hello JSP!!</h1>
10   <h1>JSP 실습입니다!!</h1>
11 </body>
12 </html>
```

클래스 파일에서 전송된 HTML 태그와 일치

3. JSP내부 객체

개발자가 객체를 생성하지 않고 바로 사용할 수 있는 객체가 내부객체이다.
JSP에서 제공되는 내부객체는 JSP컨테이너에 의해 Servlet으로 변화될 때 자동으로 객체가 생성된다.
웹 컨테이너에 의해 자동으로 구현되며 import 나 객체 선언 없이도 자유롭게 접근 가능하다.

내부 객체(자동 생성되는 객체) 종류

입출력 객체 : request, response, out
서블릿 객체 : page, config
세션 객체 : session
예외 객체 : exception

내장객체	서블릿	설명
request	javax.servlet.http.HttpServletRequest	클라이언트의 요청 정보를 저장합니다.
response	javax.servlet.http.HttpServletResponse	응답 정보를 저장합니다.
out	javax.servlet.jsp.JspWriter	JSP 페이지에서 결과를 출력합니다.
session	javax.servlet.http.HttpSession	세션 정보를 저장합니다.
application	javax.servlet.ServletContext	컨텍스트 정보를 저장합니다.
pageContext	javax.servlet.jsp.PageContext	JSP 페이지에 대한 정보를 저장합니다.
page	java.lang.Object	JSP 페이지의 서블릿 인스턴스를 저장합니다
config	javax.servlet.ServletConfig	JSP 페이지에 대한 설정 정보를 저장합니다.
exception	java.lang.Exception	예외 발생 시 예외를 처리합니다

4. 스크립트릿, 선언, 표현식

JSP문서 안에 JAVA언어를 넣기 위한 방식들 입니다. 실제 개발에서 많이 쓰이므로 잘 익혀 두자.

스크립트릿(scriptlet) : <% java 코드 기술 %>

JSP페이지에서 JAVA언어를 사용하기 위한 요소 중 가장 많이 사용되는 요소 입니다.
우리가 알고 있는 거의 모든 JAVA코드를 사용할 수 있다.

```
<body>

<%
    int i = 0;
    while(true){
        i++;
        out.println("2 * " + i + " = " + (2 * i)  + "<br />");
%>
    ==========<br />
<%
        if(i >= 9) break;
    }
%>

</body>
```

out은 내부객체(서버가 생성)임은 앞에서 이미 강의했다.

HTML코드와 JSP코드가 혼합

```
<body>
2 * 1 = 2<br />
          ==========<br />
2 * 2 = 4<br />
          ==========<br />
2 * 3 = 6<br />
          ==========<br />
2 * 4 = 8<br />
          ==========<br />
2 * 5 = 10<br />
```

scriptlet.jsp | Insert title here

http://localhost:8181/jsp_10_1_ex1_tagex/scriptlet.jsp

```
2 * 1 = 2
==========
2 * 2 = 4
==========
2 * 3 = 6
==========
2 * 4 = 8
==========
2 * 5 = 10
==========
```

Souce보기를 하면 HTML태그만 나오는 것을 알 수 있다. 서버에서 응답은 항상 HTML로 해주는 것을 잊지 말자.

위 소스를 보면 html과 스크립트릿을 이용하여 자바코드가 들어가 있음을 볼 수 있다. 하지만 웹 브라우저 상에서 소스보기를 하면 html코드만 보인다. 이미 잘 알고 있다. out객체는 내부객체이다.

4. 스크립트릿, 선언, 표현식

선언(declaration) : <%! java 코드 기술 %>

JSP페이지 내에서 사용되는 변수 또는 메서드를 선언할 때 사용 한다.
여기서 선언된 변수 및 메소드는 전역의 의미로 사용된다.

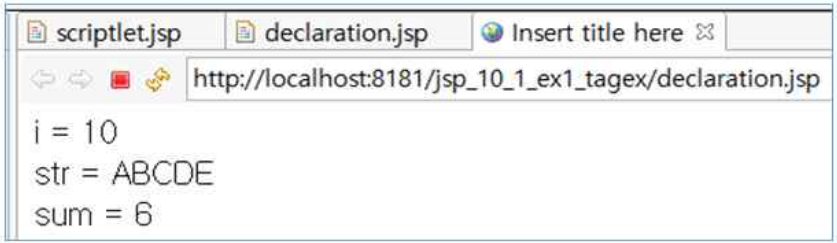
```
<body>

   <%!
      int i = 10;
      String str = "ABCDE";
   %>

   <%!
      public int sum(int a, int b) {
         return a+b;
      }
   %>

   <%
      out.println("i = " + i + "<br />");
      out.println("str = " + str + "<br />");
      out.println("sum = " + sum(1,5) + "<br />");
   %>

</body>
```



4. 스크립트릿, 선언, 표현식

표현식(expression) : <%= java 코드 기술 %>

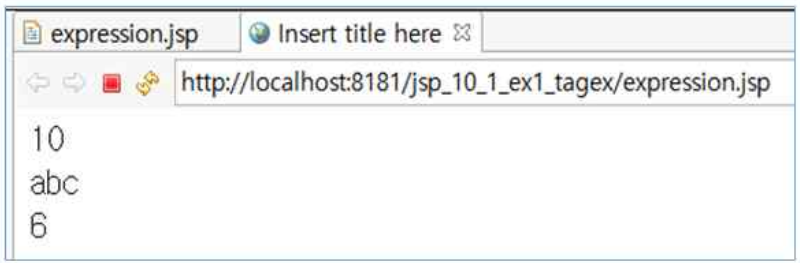
JSP페이지 내에서 사용되는 변수의 값 또는 메소드 호출 결과값을 출력하기 위해 사용된다.
결과값은 String 타입이며, ';'를 사용 하지 않는 것에 주목하자.

```
<body>
    <%!
        int i = 10;
        String str = "abc";

        private int sum(int a, int b) {
            return a+b;
        }
    %>

    <%= i %><br />
    <%= str %><br />
    <%= sum(1, 5) %>

</body>
```



감사합니다.