

# 복습하기

## 쿠키 설정

Cookie 생성자로  
쿠키 객체를 생성

표기법

Cookie 객체명 = new Cookie(이름, 값);

setMaxAge() 메서드로  
쿠키 객체의 유효기간을 설정

유효기간은 초 단위로 환산  
1주일 유효한 쿠키 : setMaxAge(7\*24\*60\*60)

표기법

객체명.setMaxAge(유효기간);

response 내장객체의  
addCookie() 메서드로  
클라이언트에 전송

클라이언트의 하드디스크에 저장

표기법

response.addCookie(객체명);

# 복습하기

## 쿠키 제거

Cookie 생성자로  
쿠키 객체를 생성

표기법

```
Cookie 객체명 = new Cookie(이름, 값);
```

setMaxAge() 메서드로  
쿠키 객체의 유효기간을 설정

유효기간을 0으로 : setMaxAge(0)

표기법

```
객체명.setMaxAge(유효기간);
```

response 내장객체의  
addCookie() 메서드로  
클라이언트에 전송

표기법

```
response.addCookie(객체명);
```

# 복습하기

## 쿠키 읽기

request.getCookie() 메서드로  
쿠키 객체를 읽어옴

`cookie[] 객체명 = request.getCookies();`

쿠키 객체의 속성 값을 반환

`쿠키객체명.getName(); // 쿠키명 반환`  
`쿠키객체명.getValue(); // 쿠키 속성값 반환`

# 복 습 하 기

## ▶ 세션 설정

### ▶ 표기: `session.setAttribute(name, value);`

#### ▶ 세션을 설정할 때, 문자형의 경우

- `String id = request.getParameter("id");`
- `session.setAttribute("id", id);`

#### ▶ 정수형일 경우

- `Integer num=new Integer(100);`
- `session.setAttribute("num", num);`

# 복습하기

## ▶ 세션 읽기

### ▶ 일반형식

▶ Object 변수명 = session.getAttribute(name);

### ▶ 문자형으로 변환하는 경우

▶ Object id\_getdata = session.getAttribute("id");

▶ String session\_id = (String)id\_getdata;

또는

▶ String session\_id = session.getAttribute("id").toString();

### ▶ 숫자형으로 변환을 하는 경우

▶ Integer num\_getdata = (Integer)session.getAttribute("num");

▶ int session\_num = num\_getdata.intValue();

## Part2. 기초 프로그래밍

- ▶ Chapter4. JSP 기본문법
- ▶ Chapter5. JSP 내장객체
- ▶ Chapter6. JSP 입력 폼 설계
- ▶ Chapter7. JSP와 DB 연동
- ▶ Chapter8. 자바빈과 액션태그
- ▶ Chapter9. 쿠키와 세션
- ▶ Chapter10. 서블릿
- ▶ Chapter11. DBCP
- ▶ Chapter12. EL
- ▶ Chapter13. JSTL

문법과  
기초 프로그래밍 실습

# 제10장 서블릿

1. 서블릿 개요
2. 서블릿의 기본구조
3. 서블릿 활용 예제
4. 컨트롤러 서블릿 작성
5. 서블릿 실행 오류와 해결방법



# 오늘의 수업 주제

서블릿 개요

서블릿의 기본구조

서블릿 활용 예제

컨트롤러 서블릿 작성

서블릿 실행 오류와 해결방법





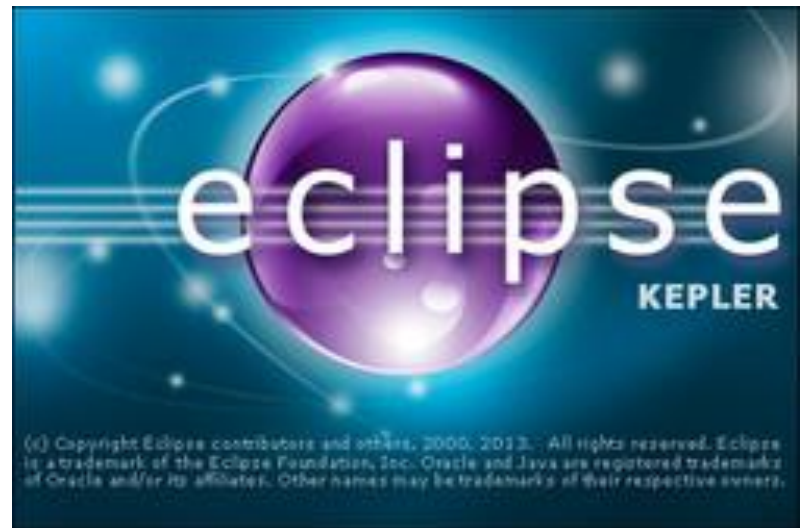
# 강의 목표

1. 서블릿 개요
2. 서블릿의 기본구조
3. 서블릿 활용 예제
4. 컨트롤러 서블릿 작성
5. 서블릿 실행 오류와 해결방법



## 10. 서블릿

1. 서블릿 개요
2. 서블릿의 기본구조
3. 서블릿 활용 예제
4. 컨트롤러 서블릿 작성
5. 서블릿 실행 오류와 해결방법

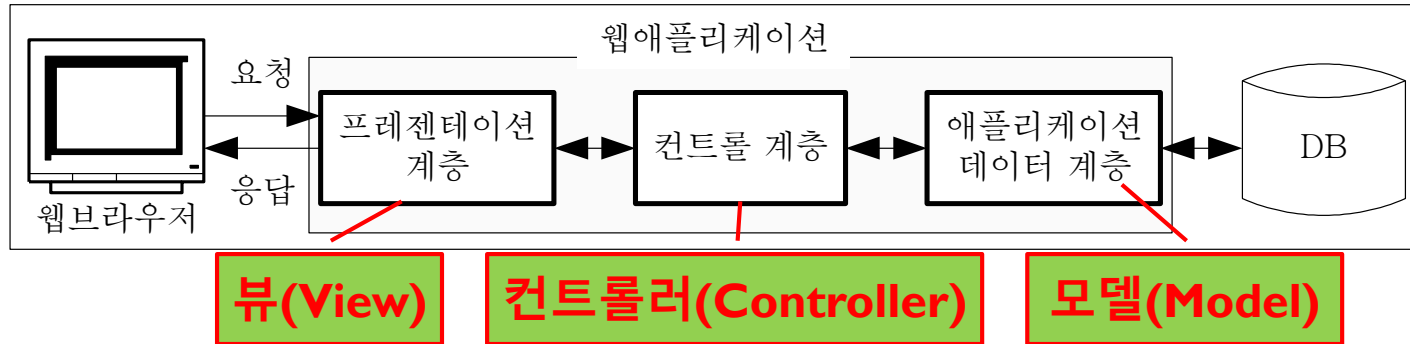


# 서블릿(Servlet) 개요

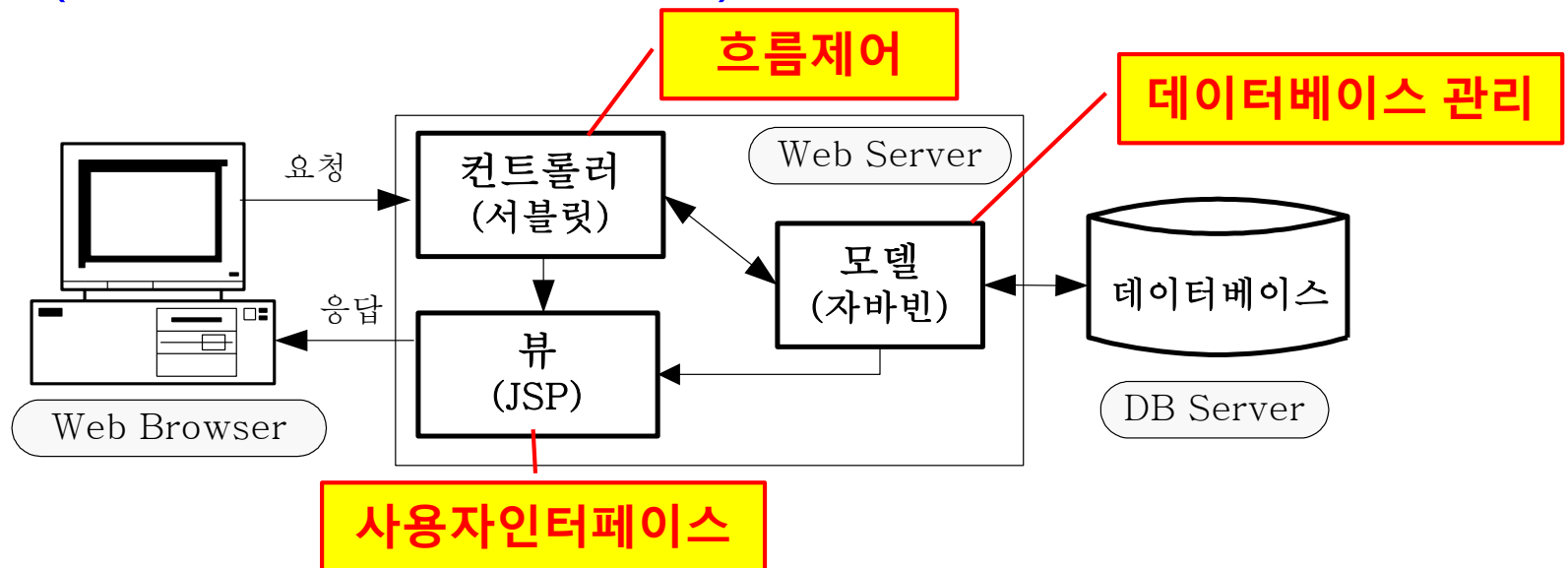


Java Servlets

## ▶ 애플리케이션의 논리적인 구조



## ▶ MVC(Model-View-Controller)



# 1. 서블릿(Servlet) 개요

## ▶ 서블릿

- ▶ 자바 플랫폼에서 컴포넌트 기반의 웹애플리케이션 개발하는 기술
- ▶ JSP는 서블릿 기술에 기반함
- ▶ 서블릿의 프리젠테이션 문제를 해결하기 위해 JSP가 등장
  - ▶ 이로 인해 웹 애플리케이션의 유지보수 어려움 발생
- ▶ JSP 모델2가 주목 받으며 다시 서블릿에 대한 중요성 부각

## ▶ 서블릿 장점

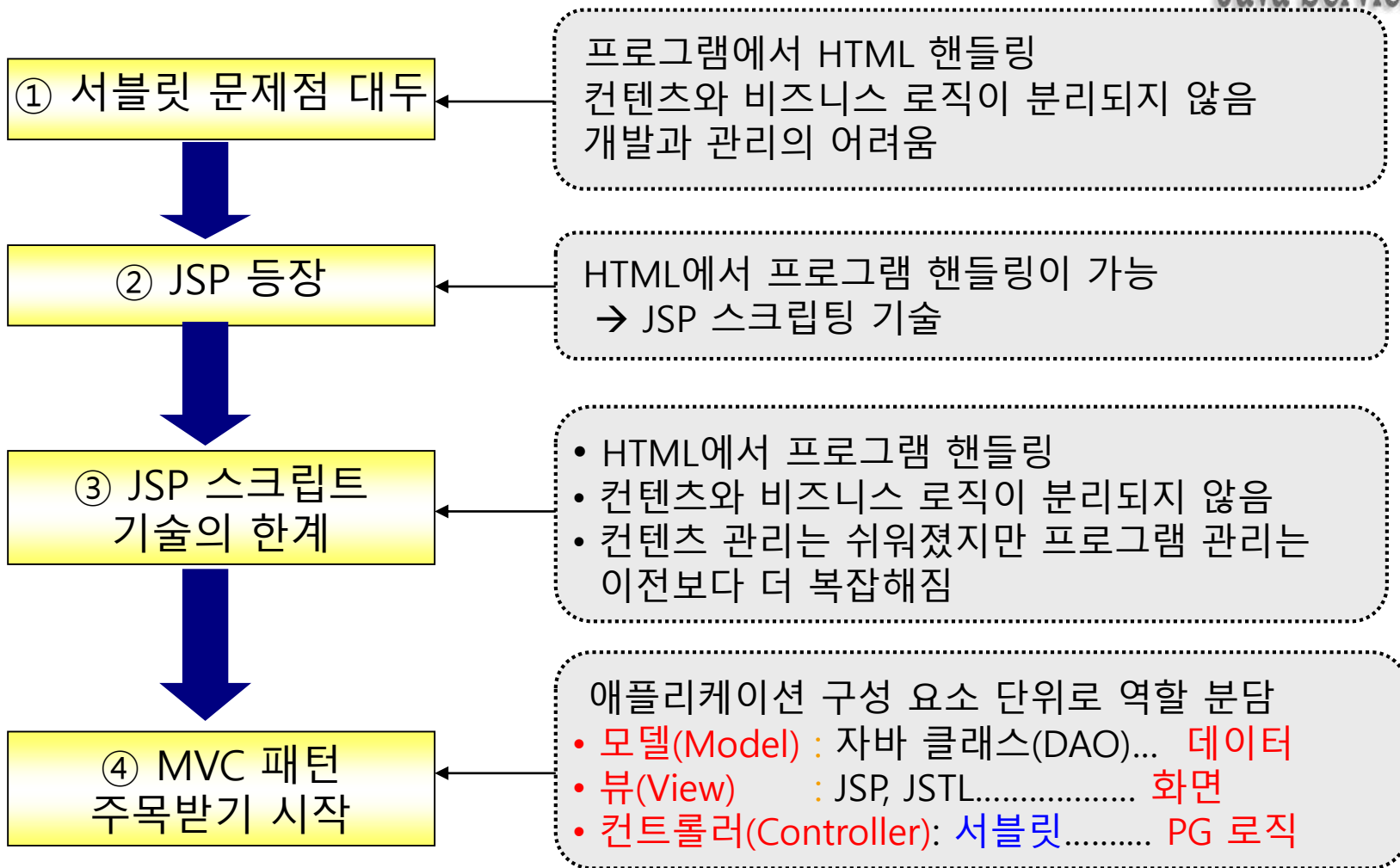
- ▶ 콘텐츠와 비즈니스 로직의 분리 가능
- ▶ 컨트롤러와 뷰의 역할 분담으로 웹디자이너와 개발자간의 공동작업 가능
- ▶ 유지보수와 기능확장 용이
- ▶ 프로그래머가 HTML, JavaScript, StyleSheet 등 모두 숙지할 필요 없음
- ▶ 쇼핑몰, DB 검색, 이미지 변환, 방명록, 게시판 등에서 이용

# 1. 서블릿 개요



Java Servlets

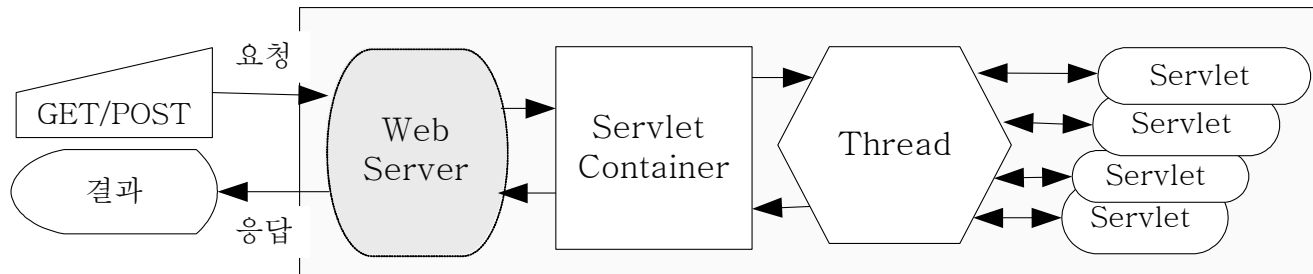
## ▶ 서블릿 변천



# 1. 서블릿 개요



## ▶ 서블릿의 처리순서



- ① 웹 브라우저에서 사용자가 서블릿 요청
- ② 웹 서버가 요청한 서블릿을 인식하여 서블릿 컨테이너에게 수행을 넘겨준다.
- ③ 서블릿 컨테이너는 스레드로 서블릿 객체를 생성하고 수행
- ④ 서블릿 실행이 종료되면 스레드가 종료되고 반환
- ⑤ 서블릿 수행결과를 웹 서버에 전송
- ⑥ 이 결과를 웹 브라우저에 전송

## 2. 서블릿의 기본구조



### ▶ 자바 클래스 형태로 구현

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

public class 서블릿클래스명 extends HttpServlet{

    public void service(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html; charset=euc-kr");
        PrintWriter out = response.getWriter();
        out.println( " 웹 브라우저로 보낼 내용");
        ...
    }
}
```

메소드명

**서블릿에 반드시 필요한 패키지**

패키지	설 명
javax.servlet	서블릿을 작성하기 위한 인터페이스와 클래스 제공
javax.servlet.http	http 프로토콜을 이용한 서블릿 작성에 필요한 인터페이스와 클래스 제공

## 2.1 HttpServlet 클래스와 서비스 메서드



- ▶ 서블릿은 클라이언트 요청에 대한 응답
  - ▶ `doGet()`과 `doPost()`, `service()` 메서드로 처리 내용 기술

메서드	설 명
<code>doGet()</code>	클라이언트가 GET 방식의 요청이 있을 때 처리
<code>doPost()</code>	클라이언트가 POST 방식의 요청이 있을 때 처리
<code>doHead()</code>	HEAD 요청을 처리
<code>doPut()</code>	PUT 방식의 요청이 있을 때 처리
<code>doDelete()</code>	DELETE 방식의 요청이 있을 때 처리
<code>doOption()</code>	OPTION 방식의 요청이 있을 때 처리
<code>service()</code>	요청의 종류와 관계없이 수행



## 2.1 HttpServlet 클래스와 서비스 메서드



### ▶ 서블릿 메서드의 기본 형식

- ▶ `void doGet(HttpServletRequest request,  
HttpServletResponse response)`
- ▶ `void doPost(HttpServletRequest request,  
HttpServletResponse response)`
- ▶ `HttpServletRequest request`
  - ▶ JSP 내장객체인 `request` 내장객체와 동일하게 클라이언트의 요청에 대한 정보를 전달받는 객체
- ▶ `HttpServletResponse response`
  - ▶ JSP 내장객체인 `response` 내장객체와 동일하게 클라이언트로 결과를 응답하고자 할 때 사용하는 객체

## 2.1 HttpServlet 클래스와 서비스 메서드

메서드	설 명
<code>getParameterNames()</code>	클라이언트의 request에 포함되어 있는 파라미터 이름을 반환
<code>getParameter(name)</code>	문자열 name의 파라미터 값을 반환
<code>getParameterValues(name)</code>	문자열 name의 파라미터 값을 배열로 반환
<code>getCookies()</code>	웹 브라우저가 전달한 쿠키 값을 가져 옴
<code>getMethod()</code>	http 요청이 get, post인지 반환
<code>getSession()</code>	현재 사용 중인 세션을 반환
<code>getRemoteAddr()</code>	클라이언트의 IP 주소를 반환
<code>getProtocol()</code>	현재 서버의 프로토콜을 문자열로 반환
<code>getCharacterEncoding()</code>	문자 데이터를 인코딩하여 반환

HttpServletRequest 클래스의 주요 메서드



## 2.1 HttpServlet 클래스와 서비스 메서드

메서드	설 명
setContentType(type)	클라이언트로 전달되는 문서의 MIME타입 설정
setHeader(name, value)	name 이름으로 value 값을 헤더 값으로 설정
setDateHeader(name, date)	name 이름으로 date에 설정된 시간 값을 헤더에 설정
sendError(status, msg)	오류 코드를 설정하고, 메시지를 보냄
setRedirect(url)	클라이언트의 요청을 다른 페이지로 보냄

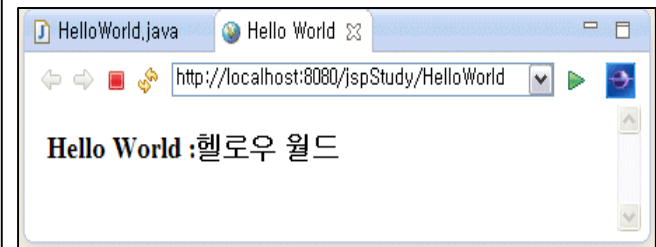
HttpServletResponse 클래스의 주요 메서드



## 2.2 간단한 서블릿 예제 프로그램

```
1 package ch10;
2
3 import java.io.PrintWriter;
4 import java.io.IOException;
5 import javax.servlet.ServletException;
6 import javax.servlet.annotation.WebServlet;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 /**
12  * Servlet implementation class HelloWorld
13  */
14 @WebServlet("/HelloWorld")
15 public class HelloWorld extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17
18     /**
19      * @see HttpServlet#HttpServlet()
20      */
21     public HelloWorld() {
22         super();
23         // TODO Auto-generated constructor stub
24     }
25
26     /**
27      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
28      */
29     protected void doGet(HttpServletRequest request,
30         HttpServletResponse response) throws ServletException, IOException {
31         // TODO Auto-generated method stub
32         response.setContentType("text/html;charset=utf-8");
33         PrintWriter out = response.getWriter();
34
35         out.println("<HTML>");
36         out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
37         out.println("<BODY><H3>Hello World :헬로우 월드</H3>");
38         out.println("</BODY></HTML>");
39     }
40
41     /**
42      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
43      */
44     protected void doPost(HttpServletRequest request,
45         HttpServletResponse response) throws ServletException, IOException {
46         // TODO Auto-generated method stub
47         doGet(request, response);
48     }
49 }
```

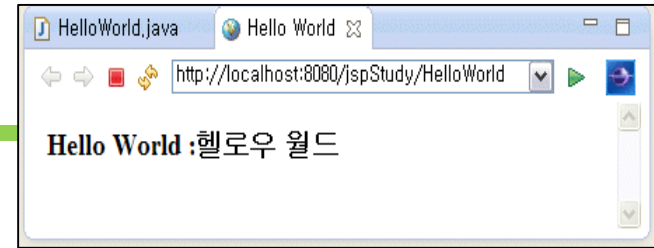
10 장 : 서블릿



실행결과



## 2.2 간단한 서블릿 예제 프로그램



### 1) 패키지(package)와 클래스 import

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

첫번째 줄에  
세가지 패키지는 반드시 선언

### 2) 서블릿 클래스 선언

- ▶ `public class 클래스명 extends HttpServlet {`
- ▶ `}`

자바 클래스 선언과 문법이 동일

### 3) doGet()메서드와 doPost() 메서드 구현

- ① `public doGet(HttpServletRequest request, HttpServletResponse response)`
- ② `throws ServletException, IOException {`
- ③ `구현내용;`
- ④ `}`
- ⑤ `public doPost(HttpServletRequest request, HttpServletResponse response)`
- ⑥ `throws ServletException, IOException {`
- ⑦ `구현내용;`
- ⑧ `}`

# 간단한 서블릿 예제 프로그램



## 4) doGet()메서드와 doPost() 메서드 구현부 작성

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

```
    response.setContentType("text/html;charset=utf-8");
```

클라이언트에 전송할 형식 지정

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

java.io.PrintWriter 클래스의 인스턴스로 처리  
- PrintWriter의 out 객체 생성

```
    out.println("<HTML>");
```

```
    out.println("<HEAD> <TITLE>Hello World</TITLE> </HEAD>");
```

```
    out.println("<BODY> <H3>Hello World : 헬로우 월드</H3>");
```

```
    out.println("</BODY> </HTML>");
```

```
}
```

```
public doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
```

```
    doGet(request, response);
```

```
}
```

Out.println()  
메서드내에  
HTML 태그로  
내용 구성

## 2.3 서블릿 예제 프로그램 입력과 실행



Java Servlets

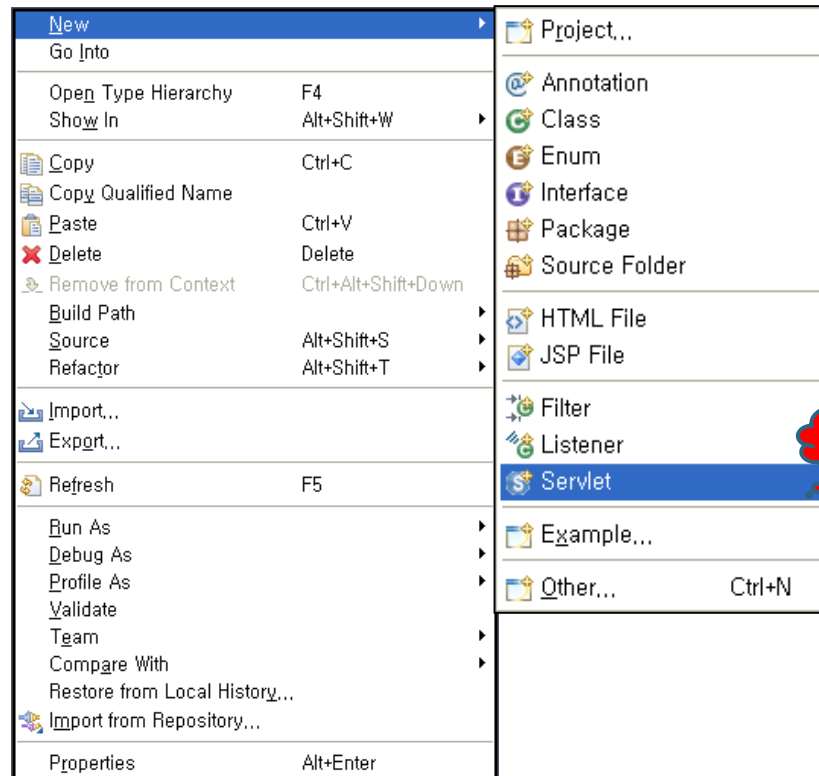
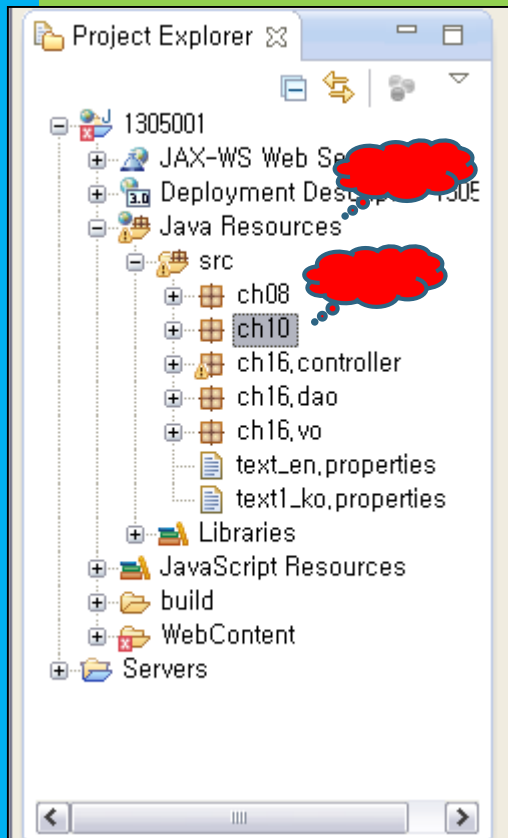
### ▶ 1) 서블릿 소스 프로그램을 위한 패키지 생성

The screenshot illustrates the steps to create a new Java package in an Eclipse IDE:

- Project Explorer:** Shows the project structure with 'jspStudy' as the root. Under 'Java Resources', the 'src' folder is highlighted.
- New Menu:** The 'New' menu is open, and 'Package' is selected from the list of options.
- New Java Package Dialog:** The dialog box is open, showing the 'Java Package' creation process. The 'Source folder' is set to 'jspStudy/src'. The 'Name' field contains 'ch10'.

10 장 : 서블릿

# 1. 입력화면 작성 및 실행





# 1. 입력화면 작성 및 실행



**Create Servlet**

Specify class file destination.

Project: jspStudy

Source folder: /jspStudy/src

Java package: ch10

Class name:

Superclass: javax.servlet.http.HttpServlet

☐ Use an existing Servlet class or JSP

Class name:

< Back Next > Finish Cancel

A red thought bubble is drawn over the "Next >" button.

HelloWorld

**Create Servlet**

Enter servlet deployment descriptor specific information.

Name: HelloWorld

Initialization parameters:

Name	Value	Description

URL mappings:

/HelloWorld

< Back Next > Finish Cancel

A red thought bubble is drawn over the "Next >" button.

# 1. 입력화면 작성 및 실행

## 서블릿소스 기본 입력화면

Create Servlet

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces:  Add...

Which method stubs would you like to create?

☒ Constructors from superclass ☒ Inherited abstract methods

☐ init ☐ destroy ☐ getServletConfig

☐ getServletInfo ☐ service ☒ doGet

☒ doPost ☐ doPut ☐ doDelete

☐ doHead ☐ doOptions ☐ doTrace

슈퍼클래스 생성자

상속된 추상메서드

```
1 package ch10;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 /**
11  * Servlet implementation class HelloWorld
12  */
13 @WebServlet("/HelloWorld")
14 public class HelloWorld extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public HelloWorld() {
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) {
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) {
36         // TODO Auto-generated method stub
37     }
38
39 }
```

# 1. 서블릿 입력화면과 소스코드 작성



Java Servlets

입력 및  
수정

```
1 package ch10;
2
3 import java.io.PrintWriter;
4 import java.io.IOException;
5 import javax.servlet.ServletException;
6 import javax.servlet.annotation.WebServlet;
7 import javax.servlet.http.*;
8
9 /**
10  * Servlet implementation class
11  */
12 @WebServlet("/HelloWorld")
13 public class HelloWorld extends HttpServlet {
14     private static final long serialVersionUID = 1L;
15
16     /**
17      * @see HttpServlet#HttpServlet()
18      */
19     public HelloWorld() {
20         // TODO Auto-generated constructor stub
21     }
22
23     /**
24      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
25      */
26     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
27         // TODO Auto-generated method stub
28         response.setContentType("text/html;charset=utf-8");
29         PrintWriter out = response.getWriter();
30
31         out.println("<HTML>");
32         out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
33         out.println("<BODY><H3>Hello World :헬로우 월드</H3>");
34         out.println("</BODY></HTML>");
35     }
36
37     /**
38      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
39      */
40     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
41         // TODO Auto-generated method stub
42         doGet(request, response);
43     }
44 }
```

# 1. 서블릿 실행

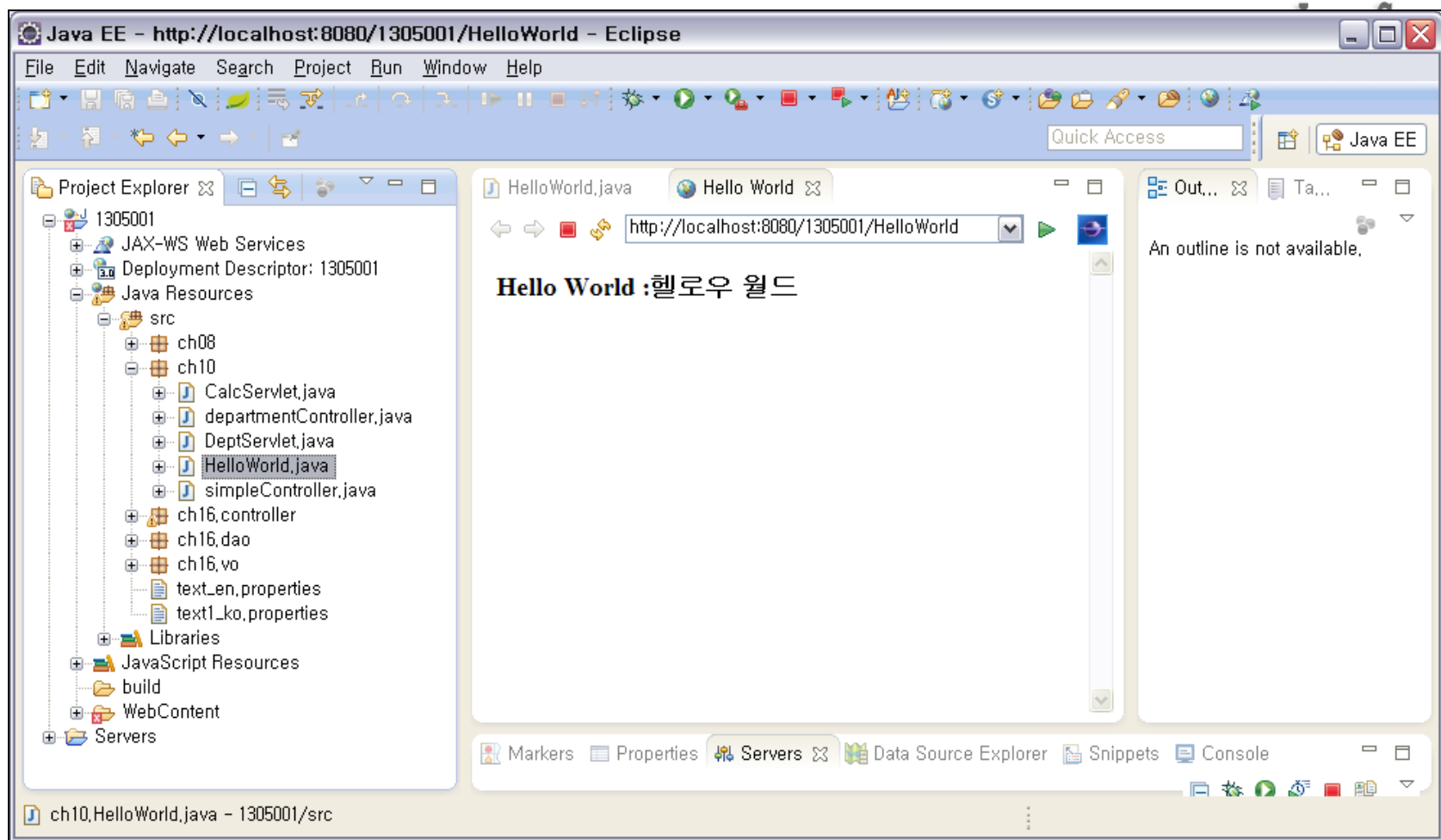


The screenshot shows the Eclipse IDE with a Java EE project named '1305001'. The Project Explorer on the left shows the project structure, including a 'src' folder with several Java files. The main editor displays the 'HelloWorld.java' file, which is a simple HTTP Servlet. A red cloud icon is placed over the 'Run' button in the toolbar.

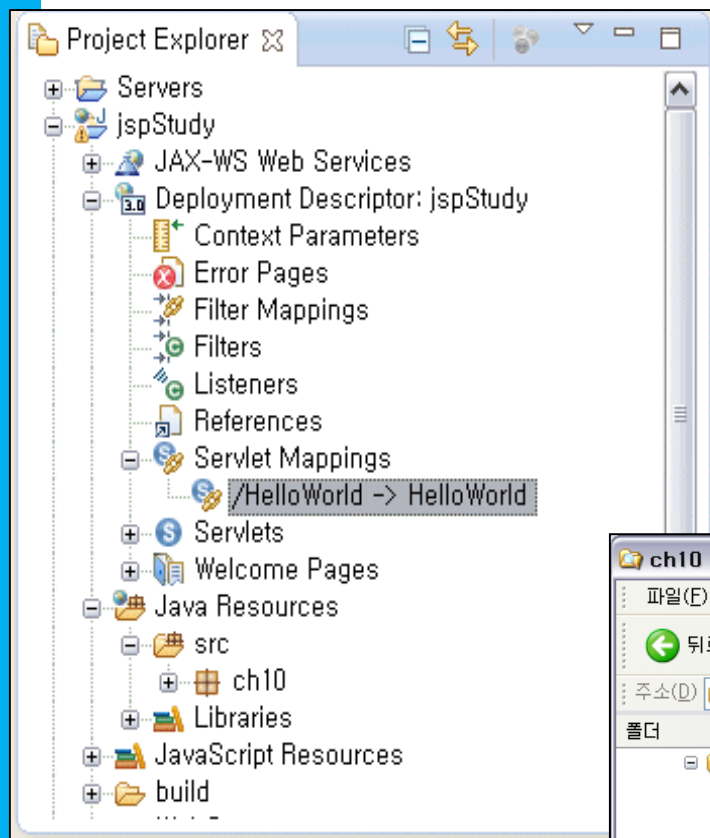
The 'Run On Server' dialog is open, showing the 'Choose an existing server' option selected. The dialog lists the available servers: 'localhost' and 'Tomcat v7.0 Server at localhost'. The 'Tomcat v7.0 Server at localhost' is currently 'Stopped'. The dialog also includes a 'Columns...' button and a checkbox for 'Always use this server when running this project'.

At the bottom of the slide, there is a green arrow pointing right, the text '28/51', and the text '10 장 서블릿'.

# 1. 서블릿 실행 결과

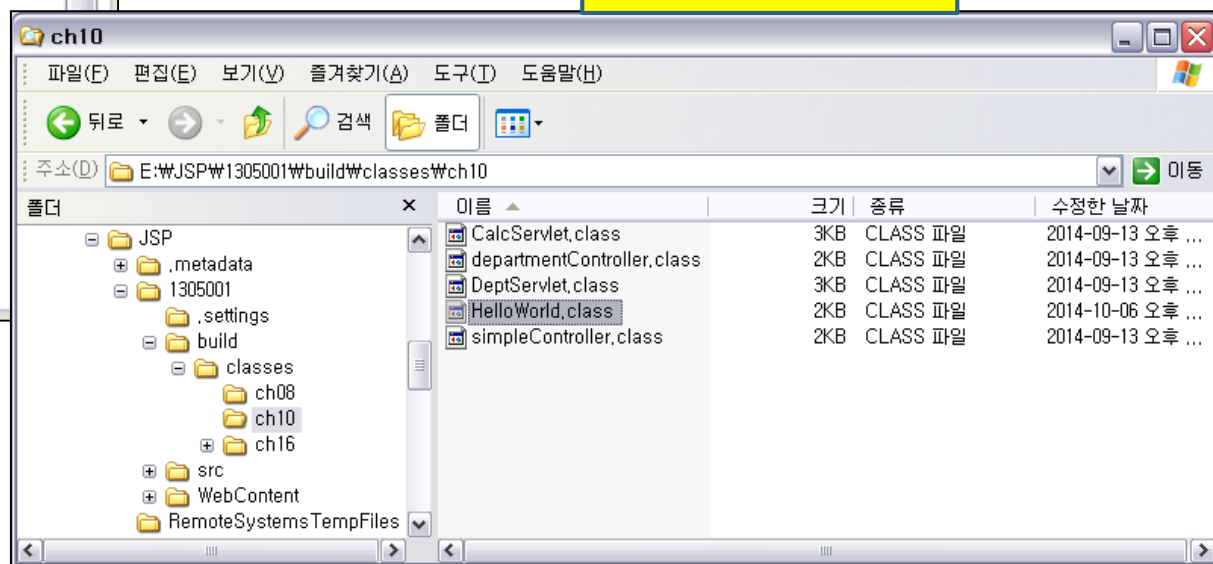


## 2. 서블릿 파일의 매핑 정보와 저장위치



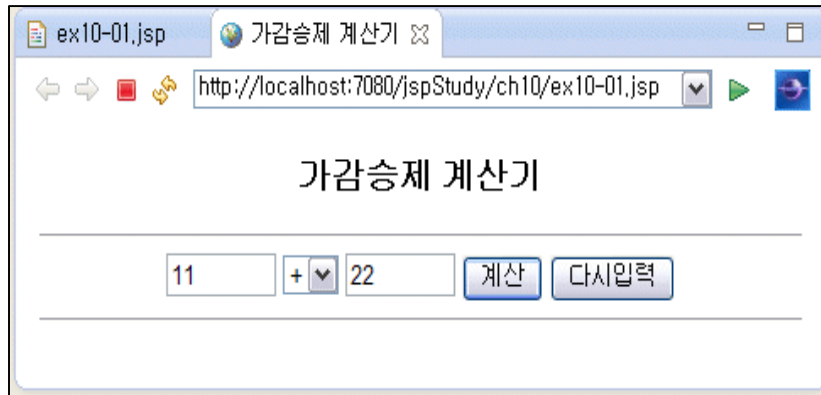
매핑정보

저장위치

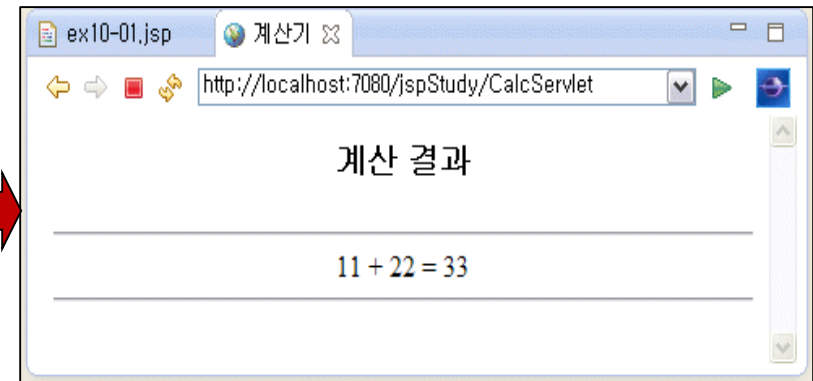


## 예제 10.1

예제 10.1의 가감승제 계산기 입력 화면에서 전송된 값으로, 서버릿의 출력 화면과 같이 계산 결과를 출력하는 가감승제 계산기 프로그램을 서버릿(CalcServlet.java)으로 작성하시오.



The screenshot shows a web browser window with the title '가감승제 계산기'. The address bar shows 'http://localhost:7080/jspStudy/ch10/ex10-01.jsp'. The page content includes the title '가감승제 계산기' and two input fields containing the numbers '11' and '22'. Between the input fields is a '+' button. To the right of the second input field are two buttons: '계산' (Calculate) and '다시입력' (Re-input).



The screenshot shows a web browser window with the title '계산기'. The address bar shows 'http://localhost:7080/jspStudy/CalcServlet'. The page content includes the title '계산 결과' (Calculation Result) and the calculation result '11 + 22 = 33' displayed on a line.





```

11 /**
12  * Servlet implementation class CalcServlet
13  */
14 @WebServlet("/CalcServlet")
15 public class CalcServlet extends HttpServlet
16     private static final long serialVersionUID
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public CalcServlet() {
21         super();
22         // TODO Auto-generated constructor stub
23     }
24     /**
25      * GET 요청을 처리하기 위한 메서드
26      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
27      */
28     protected void doGet(HttpServletRequest request,
29 HttpServletResponse response) throws ServletException,
30 // TODO Auto-generated method stub
31     doPost(request, response);
32 }

```

## CalcServlet.java



```

33
34 /**
35  * POST 요청을 처리하기 위한 메서드
36  * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
37  */
38     protected void doPost(HttpServletRequest request,
39 HttpServletResponse response) throws ServletException, IOException {
40         // TODO Auto-generated method stub
41         // 변수선언
42         int num1, num2;
43         int result;
44         String op;
45
46         // 클라이언트 응답에 대한 mime type과 문자셋 지정
47         response.setContentType("text/html; charset=utf-8");
48         // 클라이언트 응답을 위한 출력 스트림 확보
49         PrintWriter out = response.getWriter();
50
51         // HTML 폼을 통해 전송된 num1, num2 패러미터 값을 변수에 할당.
52         num1 = Integer.parseInt(request.getParameter("num1"));
53         num2 = Integer.parseInt(request.getParameter("num2"));
54         op = request.getParameter("operator");
55         // calc() 메서드 호출로 결과 받아옴.
56         result = calc(num1, num2, op);
57
58         // 출력 스트림을 통해 화면구성
59         out.println("<HTML>");
60         out.println("<HEAD><TITLE>계산기</TITLE></HEAD>");
61         out.println("<BODY><center>");
62         out.println("<H3>계산 결과</H3>");
63         out.println("<HR>");
64         out.println(num1+" "+op+" "+num2+" = "+result);
65         out.println("<HR>");
66         out.println("</BODY></HTML>");
67     }
68     // 실제 계산 기능을 수행하는 메서드
69     public int calc(int num1, int num2, String op) {
70         int result = 0;
71         if(op.equals("+")) {
72             result = num1 + num2;
73         } else if(op.equals("-")) {
74             result = num1 - num2;
75         } else if(op.equals("*")) {
76             result = num1 * num2;
77         } else if(op.equals("/")) {
78             result = num1 / num2;
79         }
80         return result;
81     }
82
83 }

```



## 예제 10.2

- ▶ 예제 10.2의 가감승제 입력화면과 입력된 값들을 서블릿 (CalcServlet.java)으로 전송하는 입력 화면을 작성하시오.

가감승제 계산기

11 + 22 계산 다시입력

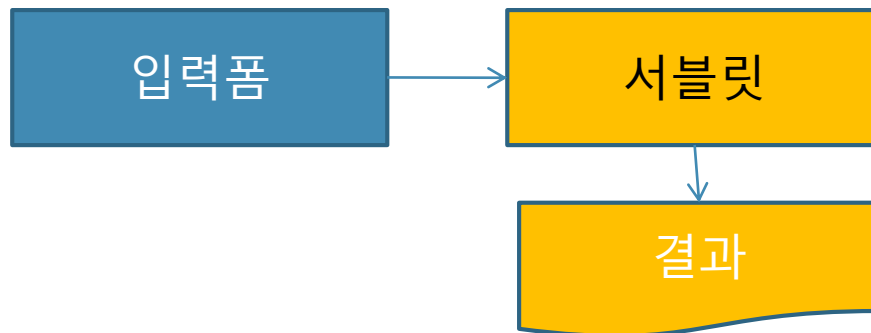
ex10-02.jsp

```
9 <body>
10 <CENTER>
11 <H3>가감승제 계산기</H3>
12 <HR>
13 <form method=post action=/jspStudy/CalcServlet name=form1>
14 <input type="text" name="num1" width=200 size="5">
15 <select name="operator">
16 <option selected>+</option>
17 <option>-</option>
18 <option>*</option>
19 <option>/</option>
20 </select>
21 <input type="text" name="num2" width=200 size="5">
22 <input type="submit" value="계산" name="b1">
23 <input type="reset" value="다시입력" name="b2">
24 </form><HR>
25 </body>
26 </html>
```

자신이 정한  
폴더명

## 예제 10.3

- ▶ 예제 10.3의 서블릿 실행 결과와 같이, 학과코드, 학과명, 전화번호를 받아서 출력하는 DeptServlet.java 프로그램을 작성하시오.



## DeptServlet.java

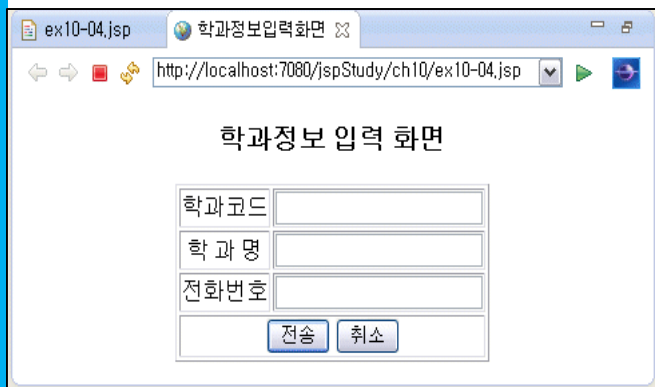


## 한글 변환방법

```
14 @WebServlet("/DeptServlet")
15 public class DeptServlet extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public DeptServlet() {
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,
29 HttpServletResponse response) throws ServletException, IOException {
30         // TODO Auto-generated method stub
31         doPost(request, response);
32     }
33
34     /**
35      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
36      */
37     protected void doPost(HttpServletRequest request,
38 HttpServletResponse response) throws ServletException, IOException {
39         // TODO Auto-generated method stub
40         response.setContentType("text/html; charset=utf-8");
41         PrintWriter out = response.getWriter();
42
43         String dept_id = request.getParameter("dept_id");
44         String dept_name = request.getParameter("dept_name");
45         String dept_tel = request.getParameter("dept_tel");
46
47         dept_id = new String(dept_id.getBytes("ISO-8859-1"), "utf-8");
48         dept_name = new String(dept_name.getBytes("ISO-8859-1"), "utf-8");
49
50         out.println("<html>");
51         out.println("<head><title>Servlet을 통한 출력 예제</title></head>");
52         out.println("<body>");
53         out.println("<h3>Servlet을 통한 출력 예제</h3>");
54         out.println("학과코드 : " + dept_id + "<p>");
55         out.println("학과명 : " + dept_name + "<p>");
56         out.println("전화번호 : " + dept_tel + "<p>");
57         out.println("</body>");
58         out.println("</html>");
59     }
60 }
```

# 예제 10.4

- ▶ 예제 10.4 학과정보 입력화면을 작성하고, DeptServlet.java 서블릿으로 전송하는 입력 화면을 작성하시오

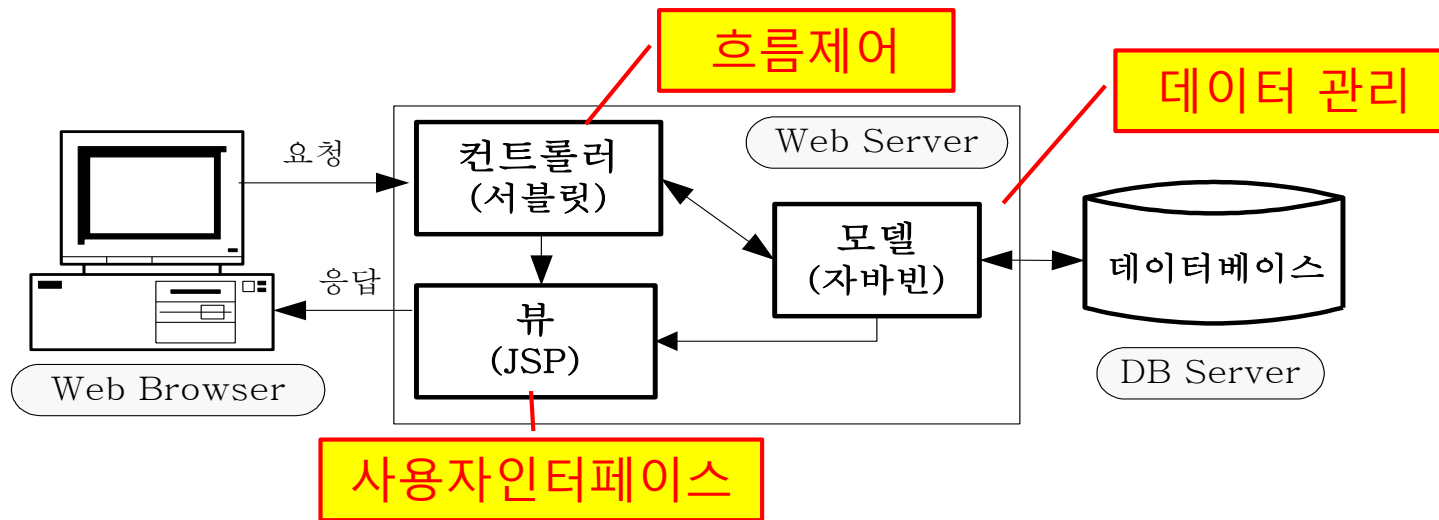


ex10-04.jsp

```
9  <body>
10  <center><h3>학과정보 입력 화면</h3>
11  <form method=post action="/jspStudy/DeptServlet">
12  <table border="1">
13      <tr>
14          <td>학과코드</td>
15          <td><input type=text name=dept_id></td>
16      </tr>
17      <tr>
18          <td>학 과 명</td>
19          <td><input type=text name=dept_name></td>
20      </tr>
21      <tr>
22          <td>전화번호</td>
23          <td><input type=text name=dept_tel></td>
24      </tr>
25      <tr>
26          <td colspan=2 align=center>
27              <input type=submit value=전송>
28              <input type=reset value=취소></td>
29      </tr>
30  </table>
31  </form></center>
32 </body>
```

학번

## 4. 컨트롤러 서블릿 작성



- ▶ 컨트롤러는 클라이언트 요청을 처리하기 위한 전체 흐름을 제어하는 역할 담당
- ▶ 다음 순서로 처리
  - ① 서블릿의 doGet() 또는 doPost() 메서드 등에서 클라이언트가 전송한 값을 검증한다.
  - ② 모델에 관한 비즈니스 로직(business logic)을 호출한다.
  - ③ 결과를 request 또는 session의 setAttribute() 메서드로 저장한다.
  - ④ 뷰로 포워딩하여 jsp 페이지로 이동한다.



## 예제 10.5

- ▶ simpleController 서블릿에 요청한 학년("year") 파라메타 값에 따라 6이면 "초등학생", 3이면 "중학생/고등학생", 4이면 "대학생"인지를 구분하는 프로그램을 작성하고, "ex10-06.jsp"로 포워딩한다.

```
14 @WebServlet("/simpleController")
15 public class simpleController extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public simpleController() {
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 Serv
29         // TODO Auto-generated method stub
30         doPost(request, response);
31     }
32
33     /**
34      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
35      */
36     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws Ser
37         // TODO Auto-generated method stub
38         String year = request.getParameter("year");
39         String result=null;
40         if (year.equals("4")) {
41             result="대학생입니다 !!!";
42         } else if(year.equals("3")) {
43             result="중학생 또는 고등학생입니다!!!";
44         } else if(year.equals("6")) {
45             result="초등학생입니다!!!";
46         } else {
47             result="알 수가 없습니다???";
48         }
49
50         request.setAttribute("result", result);
51         RequestDispatcher dispatcher = request.getRequestDispatcher("/ch10/ex10-06.jsp");
52         dispatcher.forward(request, response);
53     }
54
55 }
```



url 요청 값(year)

포워딩

## 예제 10.6

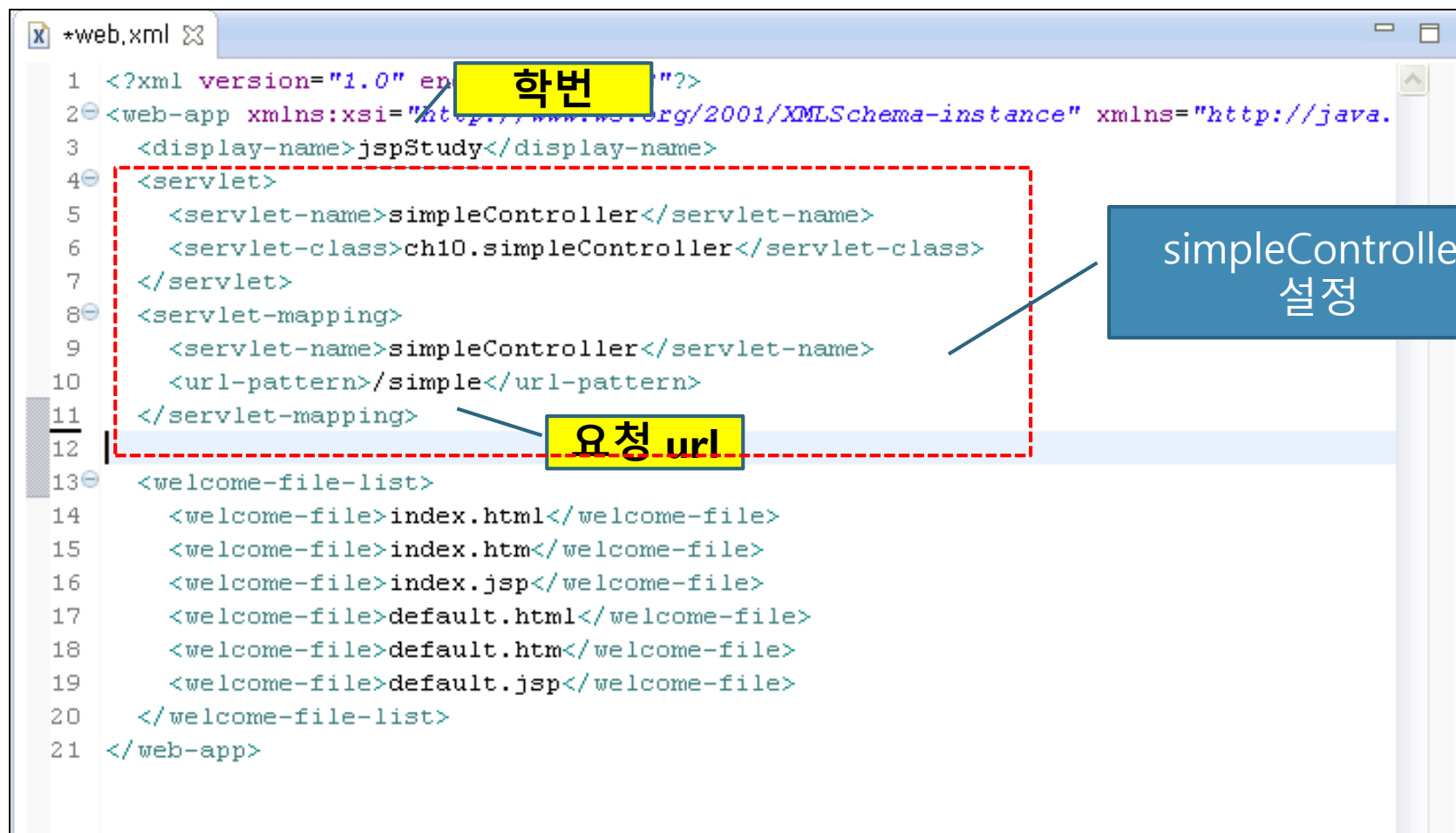
simpleController 서블릿에 저장한 값을 출력하는 뷰를 작성하시오.

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "">
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>simpleController 예제 (1)</title>
8 </head>
9 <body>
10     <h4>simpleController 예제 (1)</h4>
11     <%= request.getParameter("year") %> 년제 재학생이면, <P>
12     <%= request.getAttribute("result") %>
13 </body>
14 </html>
```

반환 값

## 예제 10.7

simpleController 서블릿의 요청 url을 "/simple"로 "web.xml" 파일에 설정하고, 아파치 탐켓을 재실행하시오.



The screenshot shows a code editor with a file named \*web.xml. The XML content is as follows:

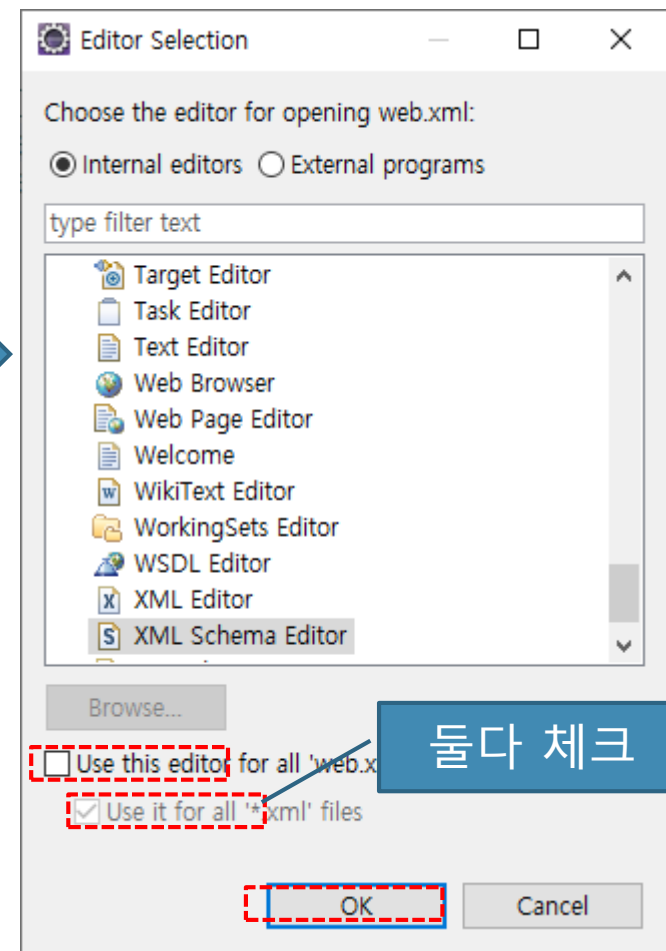
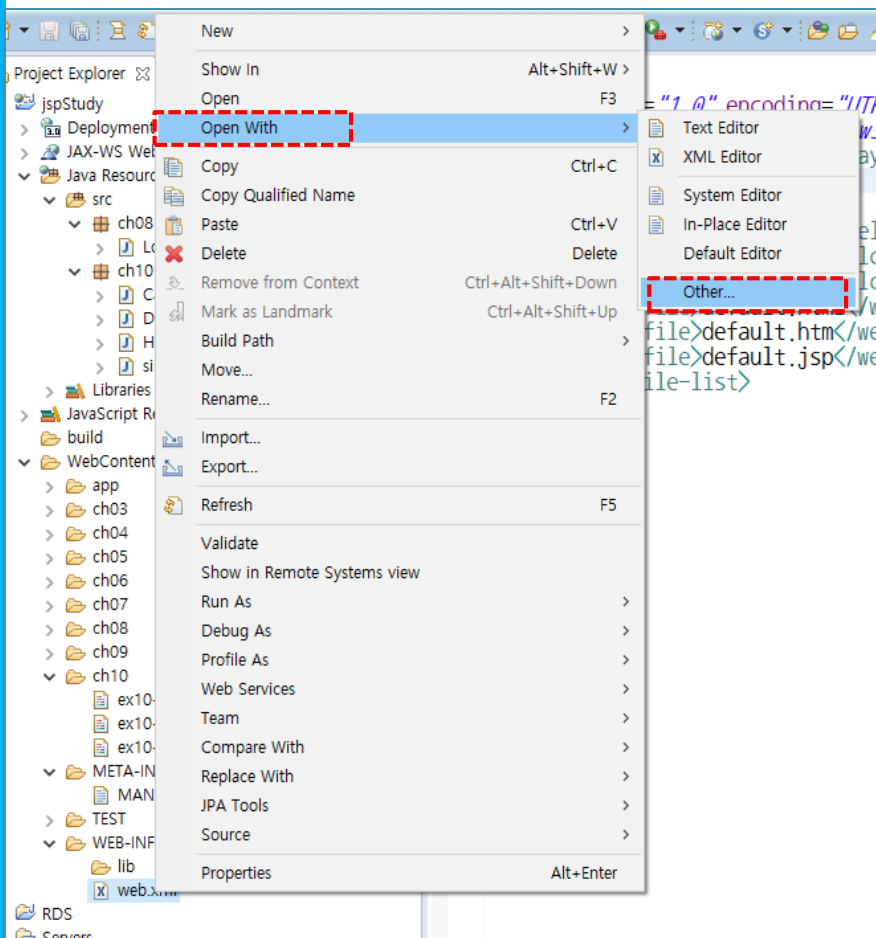
```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://www.w3.org/2001/XMLSchema-instance http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID">
3   <display-name>jspStudy</display-name>
4   <servlet>
5     <servlet-name>simpleController</servlet-name>
6     <servlet-class>ch10.simpleController</servlet-class>
7   </servlet>
8   <servlet-mapping>
9     <servlet-name>simpleController</servlet-name>
10    <url-pattern>/simple</url-pattern>
11  </servlet-mapping>
12
13  <welcome-file-list>
14    <welcome-file>index.html</welcome-file>
15    <welcome-file>index.htm</welcome-file>
16    <welcome-file>index.jsp</welcome-file>
17    <welcome-file>default.html</welcome-file>
18    <welcome-file>default.htm</welcome-file>
19    <welcome-file>default.jsp</welcome-file>
20  </welcome-file-list>
21 </web-app>
```

Annotations in the image:

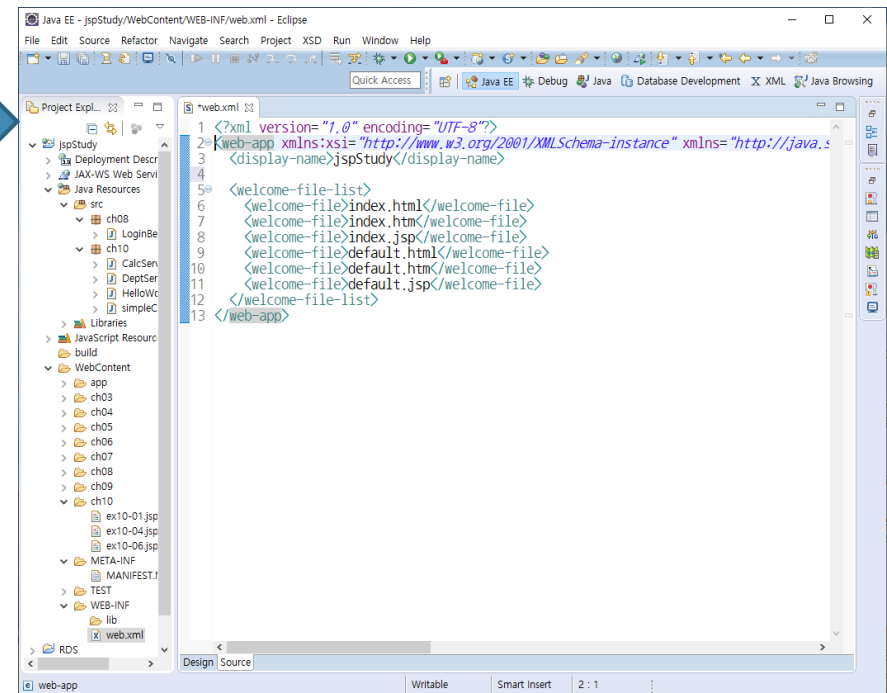
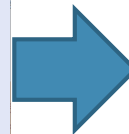
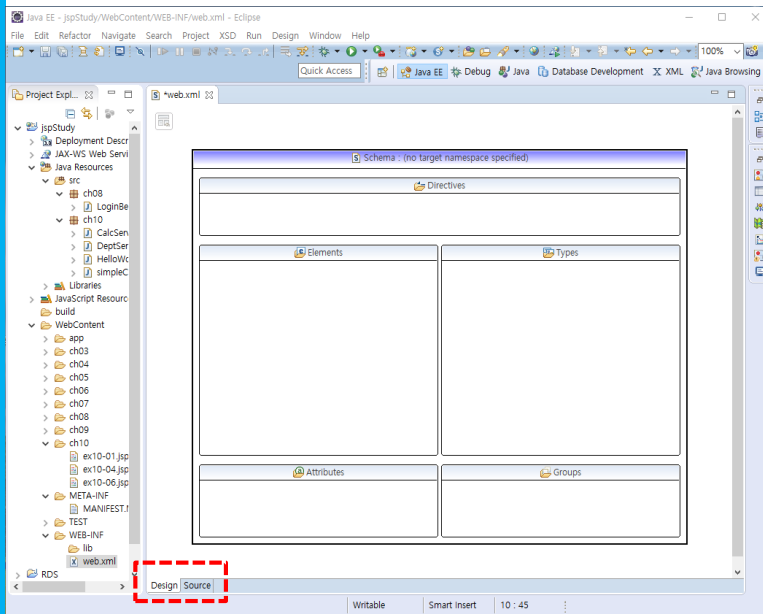
- A yellow box labeled "학번" (Student ID) is placed over the XML declaration on line 1.
- A red dashed box encloses the `<servlet>` and `<servlet-mapping>` elements from line 4 to line 11. A blue arrow points from a box labeled "simpleController 설정" (simpleController setting) to this red box.
- A yellow box labeled "요청 url" (Request url) is placed over the `<url-pattern>/simple</url-pattern>` tag on line 10, with a blue arrow pointing to it from the red dashed box.



# 이클립스 xml 파일 편집하도록 수정(1)



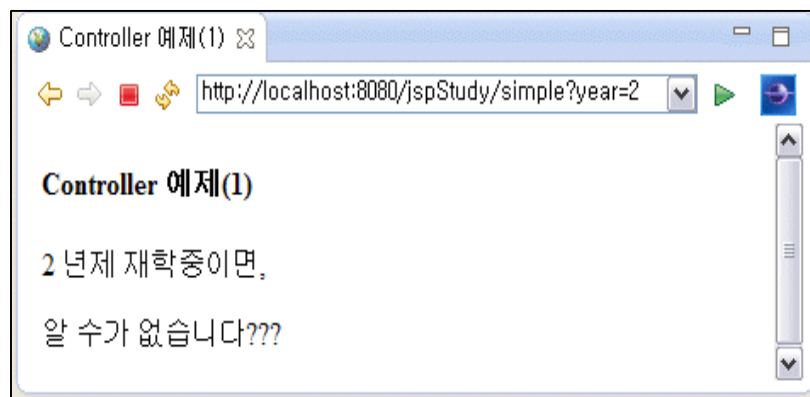
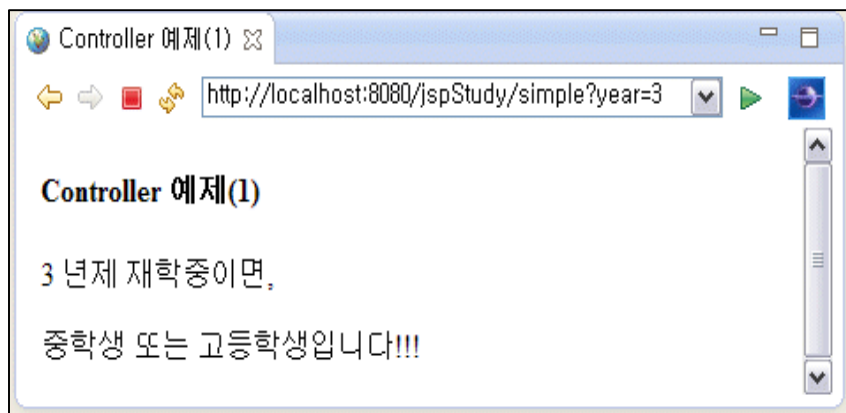
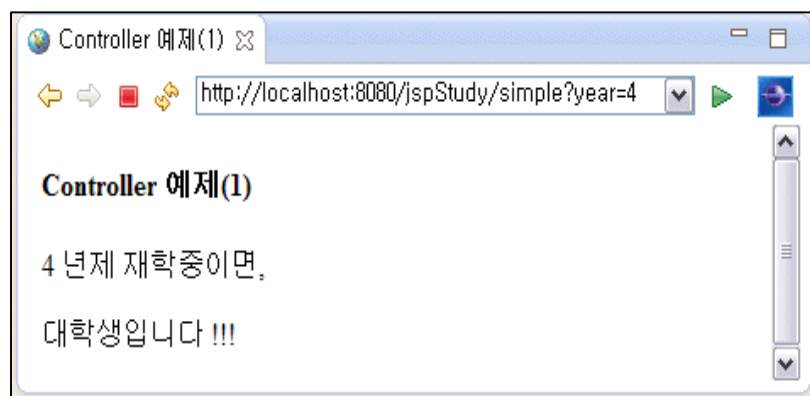
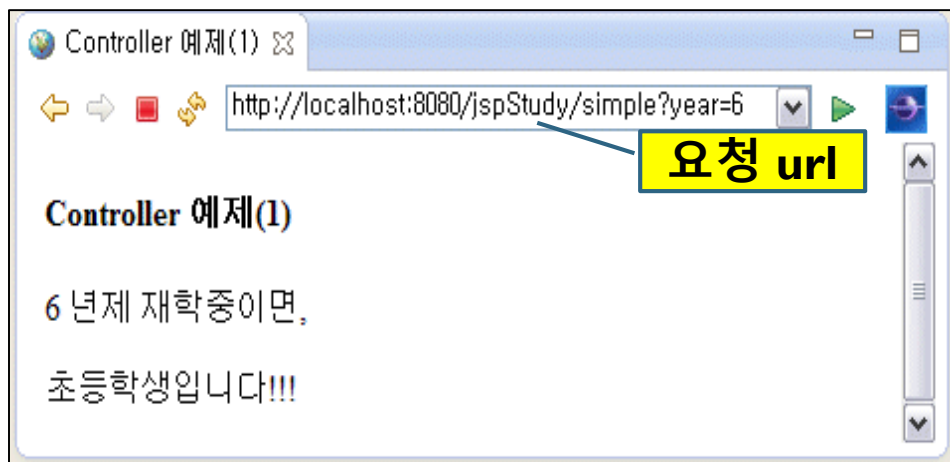
# 이클립스 xml 파일 편집하도록 수정[2]



## 예제 10.8

실행 결과와 같이 예제 10.5부터 예제 10.7에서 작성 내용을 토대로 year 파라메타 값을 6, 3, 4, 2로 지정하여 실행하시오.  
단, "http://localhost:8080/학번/simple?year=값" 임

실행방법



## 예제 10.9

파라메타("type")가 널(null) 또는 "select"이면 "./ch07/ex07-01.jsp", "insert"이면 "./ch07/ex07-02.jsp", "update"이면 "./ch07/ex07-03.jsp", "delete"이면 "./ch07/ex07-04.jsp"로 포워딩하는 departmentController를 작성하시오.

```
14 @WebServlet("/departmentController")
15 public class departmentController extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public departmentController() {
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,
29 HttpServletResponse response) throws ServletException, IOException {
30         // TODO Auto-generated method stub
31         doPost(request, response);
32     }
33
34     /**
35      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
36      */
37     protected void doPost(HttpServletRequest request,
38 HttpServletResponse response) throws ServletException, IOException {
39         // TODO Auto-generated method stub
40         String type = request.getParameter("type");
41         String path="";
42         if(type == null || type.equals("select")) {
43             path="./ch07/ex07-01.jsp";
44         } else if(type.equals("insert")) {
45             path="./ch07/ex07-02.jsp";
46         } else if(type.equals("update")) {
47             path="./ch07/ex07-03.jsp";
48         } else {
49             path="./ch07/ex07-04.jsp";
50         }
51         RequestDispatcher dispatcher = request.getRequestDispatcher(path);
52         dispatcher.forward(request, response);
53     }
54 }
```



## 예제 10.10

departmentController를 "web.xml" 파일에 요청 <uri-pattern>으로 "/Department"로 추가 설정하고, 아파치 탐켓을 재실행하여 type을 select로 지정하여 실행하시오. 단, insert, update, delete인 경우 이 방법으로는 실행이 불가함.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
3   <display-name>jspStudy</display-name>
4   <servlet>
5     <servlet-name>simpleController</servlet-name>
6     <servlet-class>ch10.simpleController</servlet-class>
7   </servlet>
8   <servlet-mapping>
9     <servlet-name>simpleController</servlet-name>
10    <url-pattern>/simple</url-pattern>
11  </servlet-mapping>
12
13  <servlet>
14    <servlet-name>departmentController</servlet-name>
15    <servlet-class>ch10.departmentController</servlet-class>
16  </servlet>
17  <servlet-mapping>
18    <servlet-name>departmentController</servlet-name>
19    <url-pattern>/Department</url-pattern>
20  </servlet-mapping>
21
22  <welcome-file-list>
23    <welcome-file>index.html</welcome-file>
24    <welcome-file>index.htm</welcome-file>
25    <welcome-file>index.jsp</welcome-file>
26    <welcome-file>default.html</welcome-file>
27    <welcome-file>default.htm</welcome-file>
28    <welcome-file>default.jsp</welcome-file>
29  </welcome-file-list>
30 </web-app>
```

추가함

- ▶ 실행방법 : <http://localhost:8080/학번/Department?type=select>

검색만 실행됨

테이블검색

http://localhost:8080/jspStudy/Department?type=select

[[ Department 테이블 검색 ]]

순번	학과코드	학 과 명	전화번호
1	컴공	컴퓨터계열	765-7777
2	정통	정보통신공학과	765-4200
3	경영	경영학과	765-4400
4	행정	세무행정학과	765-4500

DB에서 정상적으로 검색 되었습니다!!!

- ▶ 실행방법 : <http://localhost:8080/학번/Department?type=insert>

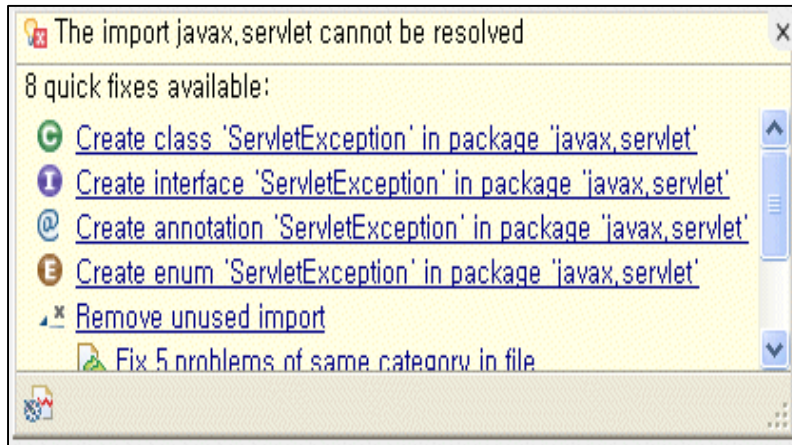
Servlet Mappings

/CalcServlet	-> CalcServlet
/courseController	-> courseController
/Department	-> departmentController
/departmentController	-> departmentController
/DeptServlet	-> DeptServlet
/HelloWorld	-> HelloWorld
/simple	-> simpleController
/simpleController	-> simpleController

서블릿 매핑 목록

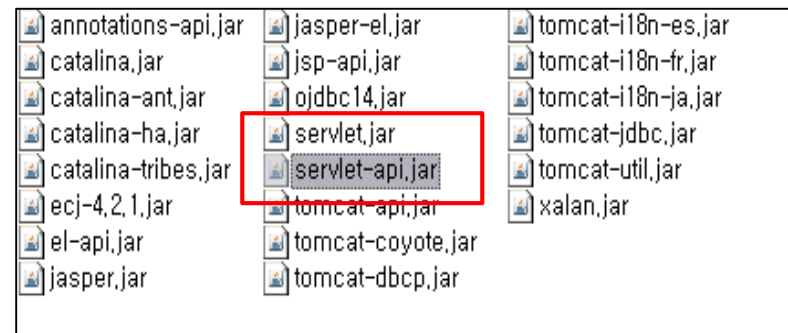
# 5. 서블릿 실행 오류와 해결 방법

## ▶ 5.1 서블릿 실행 관련 파일이 없을 때



### servlet.jar, servlet-api.jar 파일

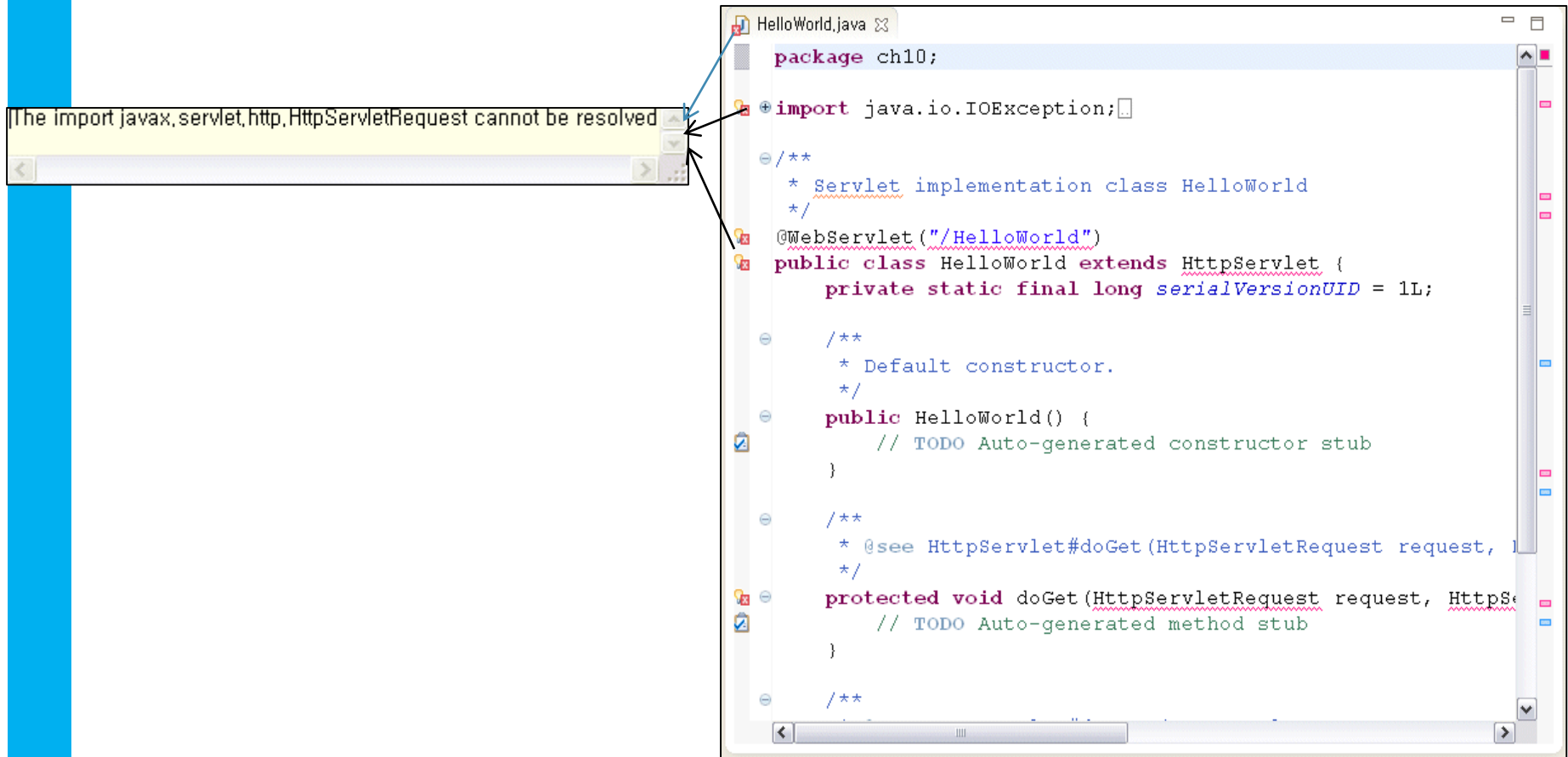
아파치 탐켓이 설치된 "lib" 폴더 또는 jdk가 설치된 jre의 "lib/ext" 폴더에 없거나 path로 지정되지 않으면 "The import javax.servlet cannot be resolved" 오류가 발생



# 5. 서블릿 실행 오류와 해결 방법



## ▶ 5.2 이클립스에서 서블릿 소스 코드의 오류 표시



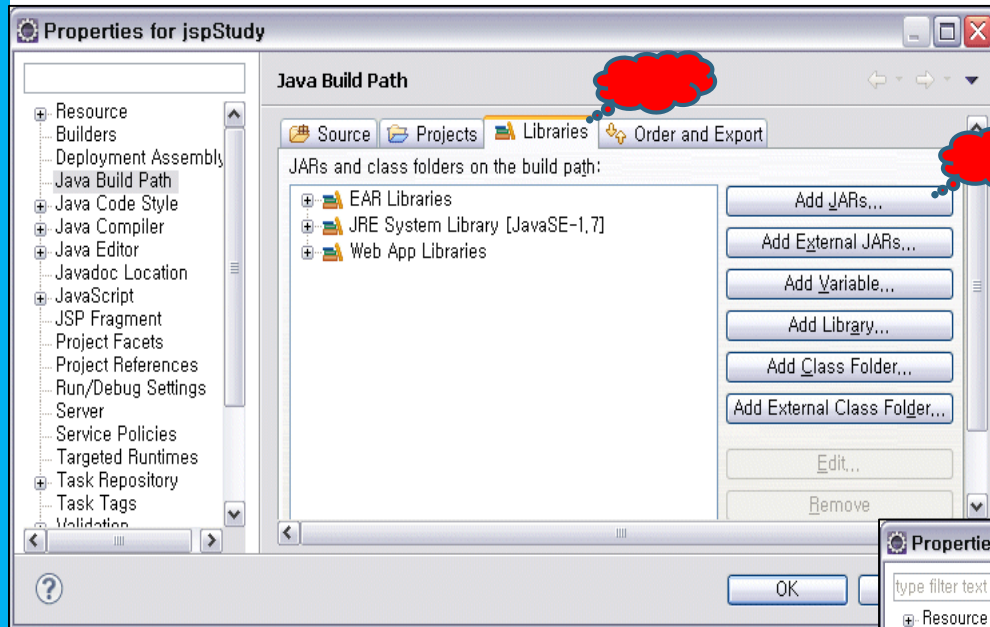


# 5. 서블릿 실행 오류와 해결 방법

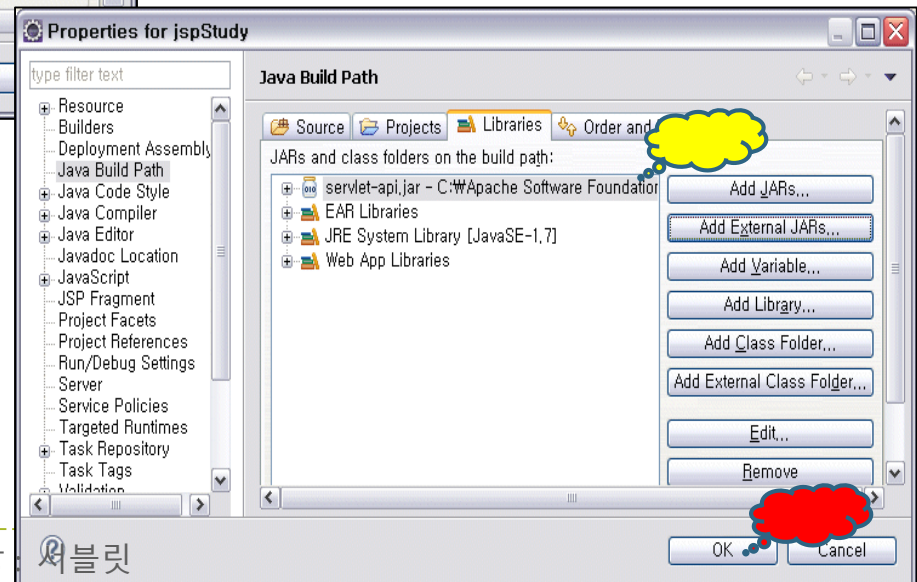


Java Servlets

## ▶ 5.2 이클립스에서 서블릿 소스 코드의 오류 표시 해결



servlet-api.jar 추가



# 강 의 내 용 요약 정 리

## ▶ 서블릿의 기본구조 : 자바클래스 형태로 구현

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

public class 서블릿클래스명 extends HttpServlet{

    public void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println( " 웹 브라우저로 보낼 내용");
        ...
    }
}
```



Java Servlets

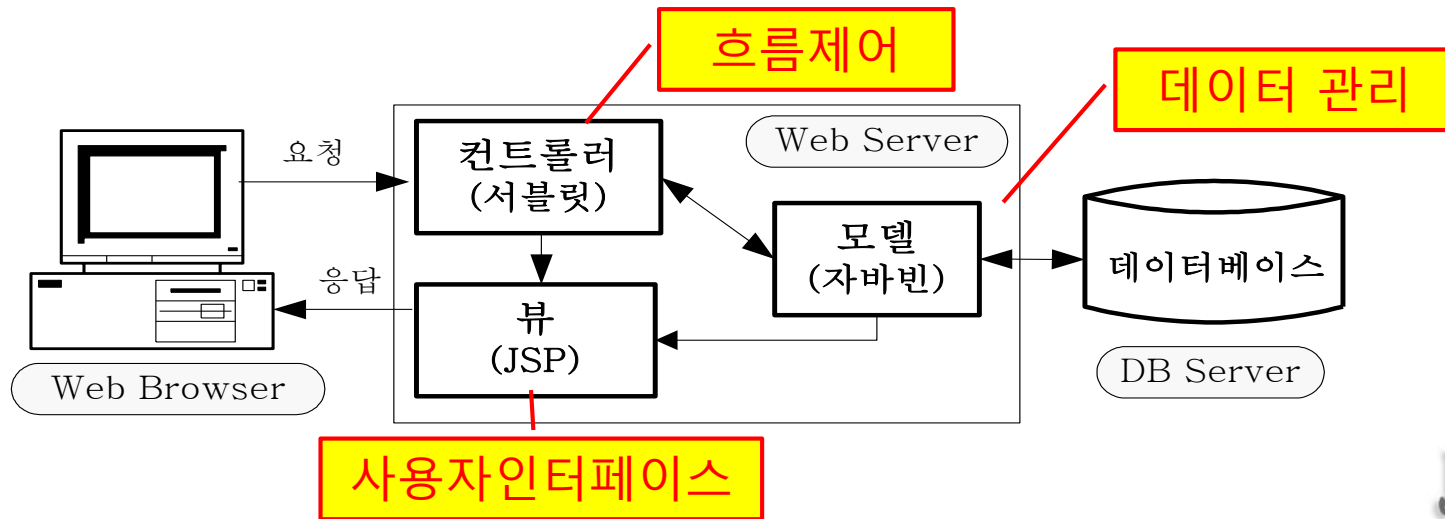
# 강 의 내 용 요약 정 리

- ▶ 서블릿은 클라이언트 요청에 대한 응답
  - ▶ doGet()과 doPost(), service() 메서드로 처리 내용 기술

메서드	설 명
doGet()	클라이언트가 GET 방식의 요청이 있을 때 처리
doPost()	클라이언트가 POST 방식의 요청이 있을 때 처리
doHead()	HEAD 요청을 처리
doPut()	PUT 방식의 요청이 있을 때 처리
doDelete()	DELETE 방식의 요청이 있을 때 처리
doOption()	OPTION 방식의 요청이 있을 때 처리
service()	요청의 종류와 관계없이 수행



# 강 의 내 용 요약 정 리

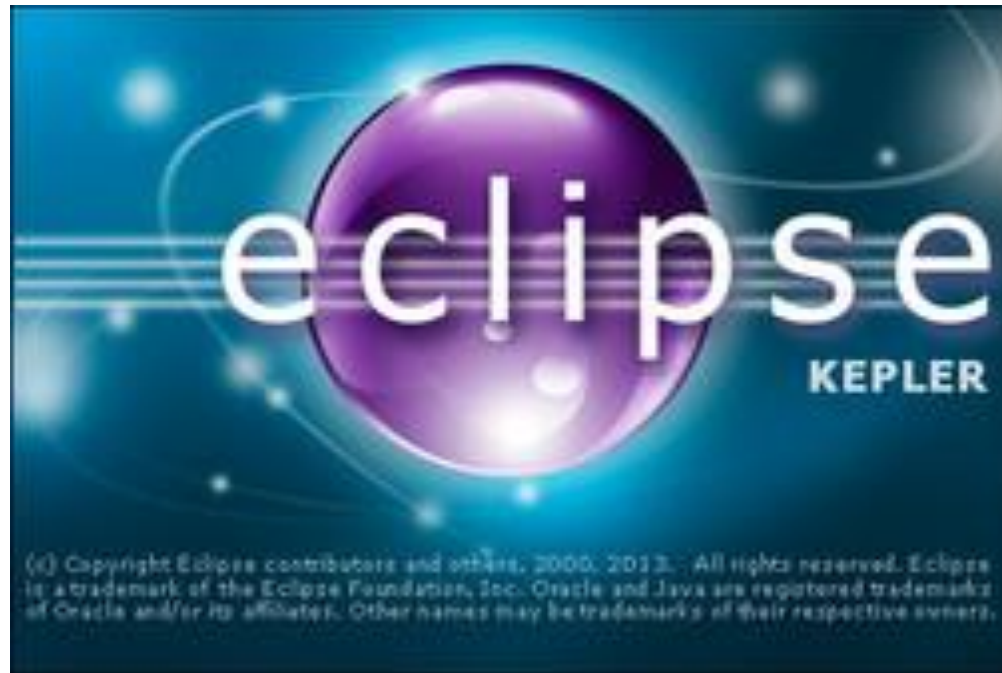


- ▶ 컨트롤러는 클라이언트 요청을 처리하기 위한 전체 흐름을 제어하는 역할 담당
- ▶ 다음 순서로 처리
  - ① 서블릿의 doGet() 또는 doPost() 메서드 등에서 클라이언트가 전송한 값을 검증한다.
  - ② 모델에 관한 비즈니스 로직(business logic)을 호출한다.
  - ③ 결과를 request 또는 session의 setAttribute() 메서드로 저장한다.
  - ④ 뷰로 포워딩하여 jsp 페이지로 이동한다.



장 주제

## 11장. DBCP



(c) Copyright Eclipse contributors and others, 2000, 2013. All rights reserved. Eclipse is a trademark of the Eclipse Foundation, Inc. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.