

3

Servlet의 핵심 사항들



이 장에서 다룰 내용

1

클라이언트에서 서블릿으로 요청하는 방식

2

서블릿을 이용한 클라이언트에서 전송되는
요청처리

3

서블릿에서 한글 처리하기

4

하나의 파라미터 이름으로 여러 개의
파라미터 값 처리하기

5

서블릿에서 세션 살펴보기



클라이언트에서 서블릿으로 요청하는 방식

□ GET 방식

- 사용 방식 : `[2]`
- GET 방식으로 요청이 전송되는 경우
 - 브라우저 주소 표시줄에 주소를 직접 입력해서 요청을 전송하는 경우
 - Html의 a 태그를 사용해서 링크를 걸어서 전송하는 경우
`목록보기`
 - Html 폼 태그에서 method 속성을 GET으로 지정하는 경우
`<form action="" name="" method="GET">`



[그림 3-1] GET 방식의 요청 처리



클라이언트에서 서블릿으로 요청하는 방식

□ POST 방식

- 사용 방식 : `<form name=" " action=" " method="POST">`
- 회원 가입 요청, 게시판 글쓰기 요청, 자료실 업로드 등을 처리할 때 사용하는 방식



[그림 3-2] POST 방식의 요청 처리



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

□요청이 GET 방식으로 전송되어 올 경우

■ 클라이언트 페이지 생성하기

• login.html

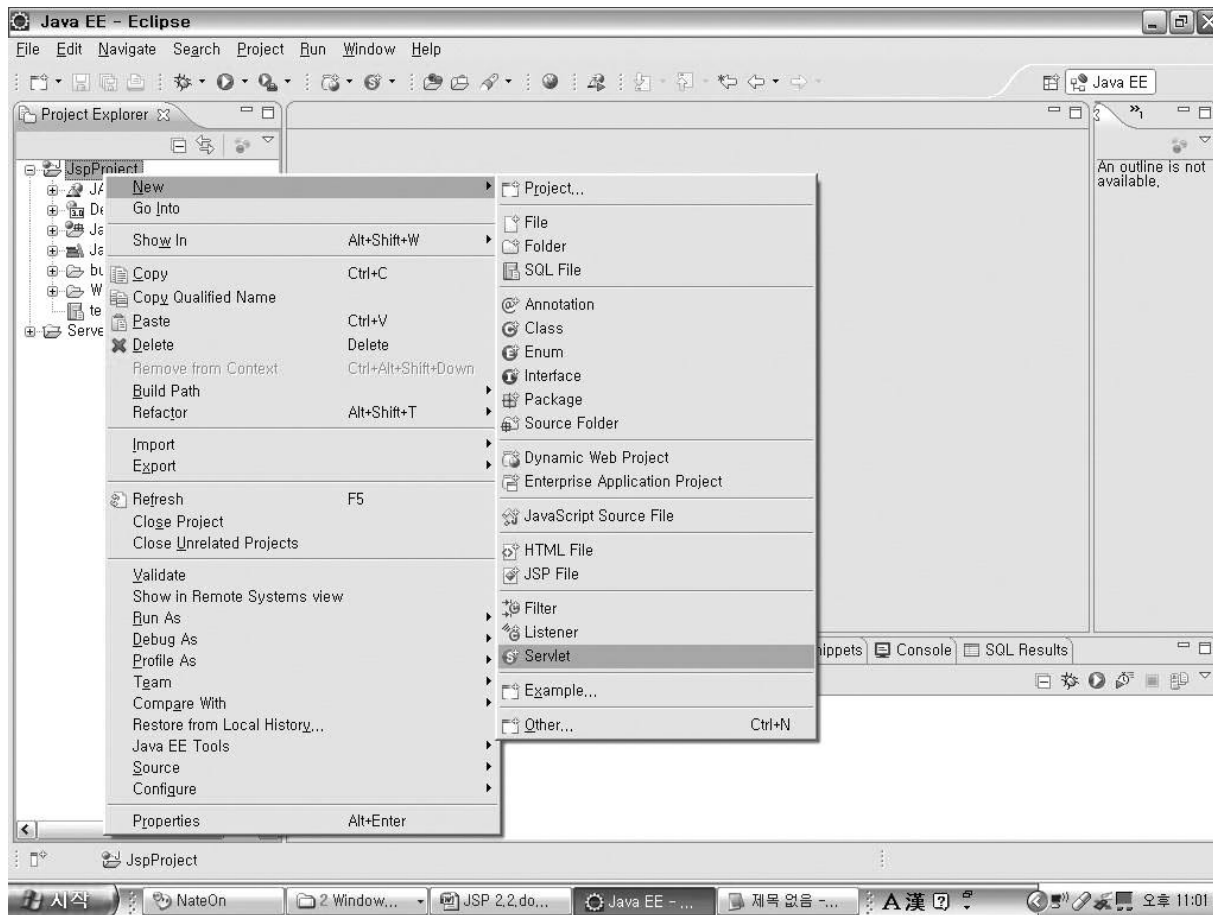
```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
2  html4/loose.dtd">
3  <html>
4  <head>
5  <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
6  <title>Insert title here</title>
7  </head>
8  <body>
9  <h1>로그인</h1>
10 <form action="login" method="get">
11 아이디 : <input type="text" name="id"/><br>
12 비밀번호 : <input type="text" name="passwd"><br>
13 <input type="submit" value="로그인"/>
14 </form>
15 </body>
16 </html>
```



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

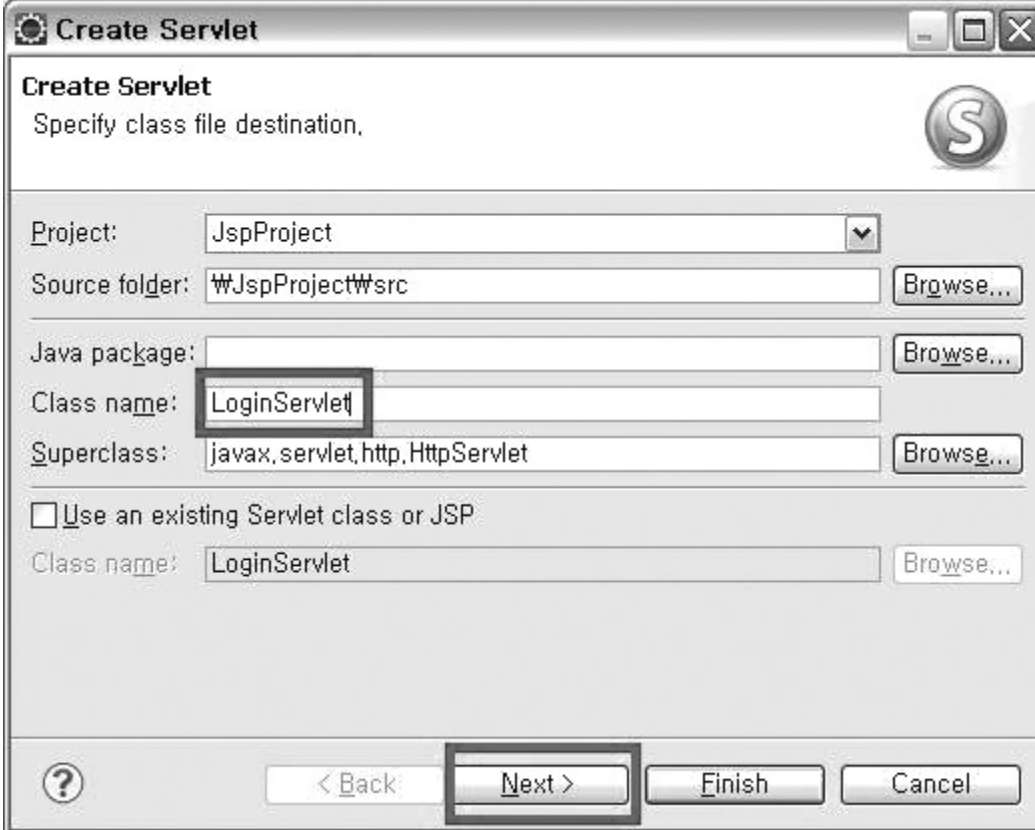
■ 서블릿 생성하기

• 01. [New]-[Servlet]을 클릭



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

- 02. Class name에 LoginServlet을 입력하고 [Next] 버튼을 클릭



The image shows a 'Create Servlet' dialog box from an IDE. The 'Class name' field is highlighted with a black box and contains the text 'LoginServlet'. The 'Next >' button at the bottom is also highlighted with a black box. Other fields include 'Project' (JspProject), 'Source folder' (WJspProject\src), 'Java package' (empty), 'Superclass' (javax.servlet.http.HttpServlet), and 'Use an existing Servlet class or JSP' (unchecked).

Create Servlet
Specify class file destination.

Project: JspProject

Source folder: WJspProject\src **Browse...**

Java package: **Browse...**

Class name: **LoginServlet**

Superclass: javax.servlet.http.HttpServlet **Browse...**

☐ Use an existing Servlet class or JSP

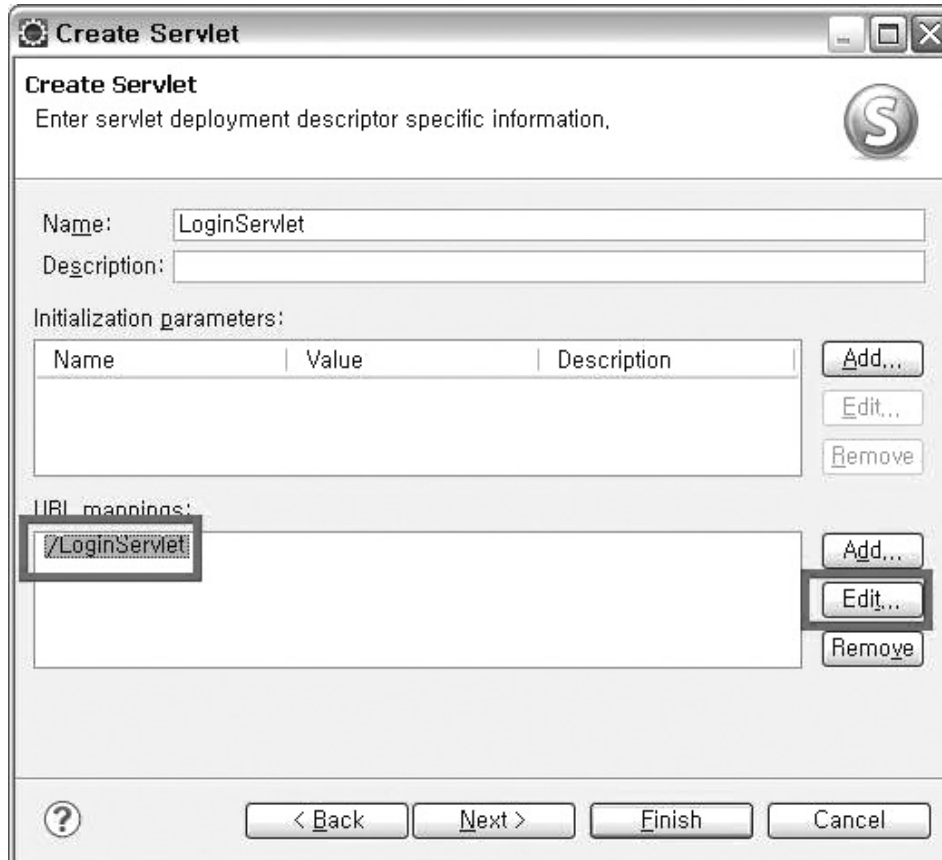
Class name: LoginServlet **Browse...**

Next > **Finish** **Cancel**



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

- 03. 다음 화면에서 URL mappings 항목을 선택하고 [Edit] 버튼을 클릭



The image shows a 'Create Servlet' dialog box with the following fields and buttons:

- Name:** LoginServlet
- Description:** (empty)
- Initialization parameters:** (empty table with columns Name, Value, Description)
- URL mappings:** /LoginServlet (highlighted with a red box)

Buttons on the right side of the dialog:

- Add...
- Edit.. (highlighted with a red box)
- Remove

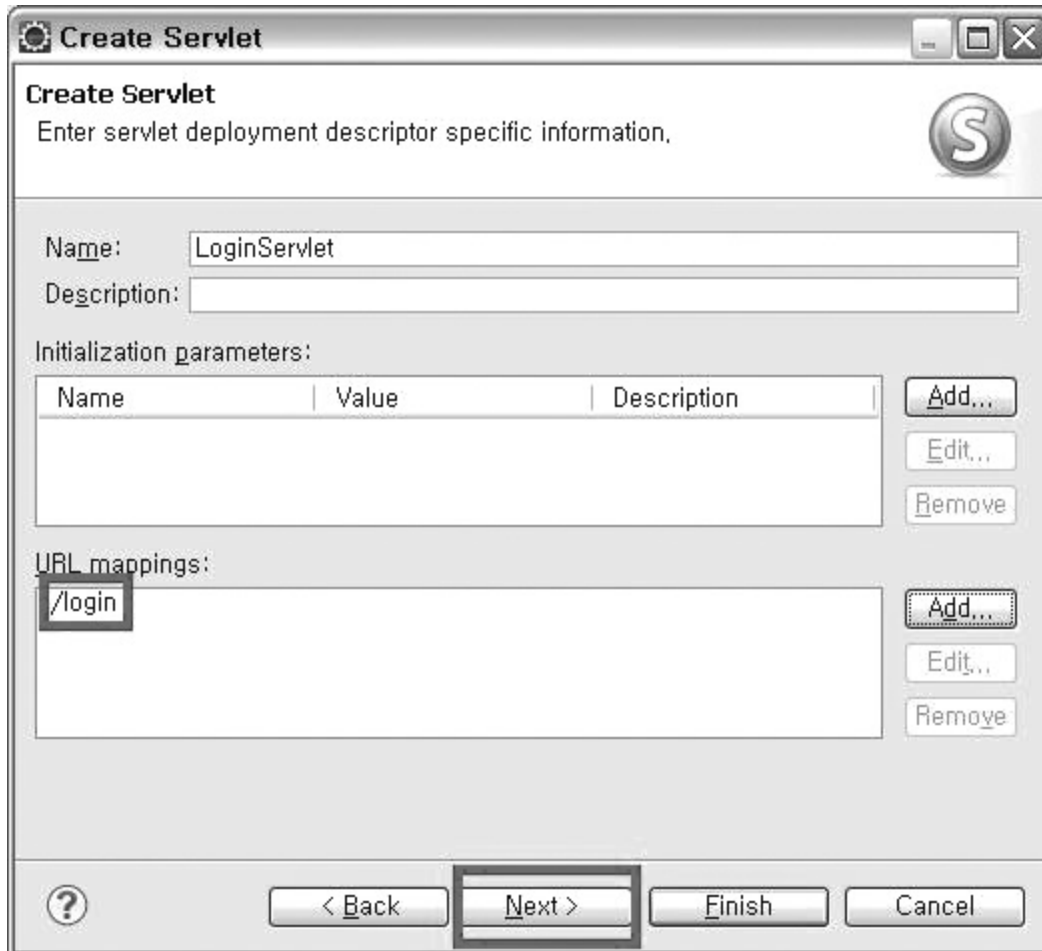
Buttons at the bottom:

- < Back
- Next >
- Finish
- Cancel



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

- 04. 클라이언트 폼 태그의 요청 경로가 login(<form action= "login" >)으로 설정되어 있으므로 URL mappings 값을 /login으로 수정한다. 이 부분은 web.xml 설정 파일에서 <url-pattern> 항목의 내용으로 추가되는 부분이다.



The image shows a 'Create Servlet' dialog box from an IDE. It contains fields for 'Name' (LoginServlet) and 'Description'. Below these are 'Initialization parameters' and 'URL mappings'. The 'URL mappings' field contains '/login', which is highlighted with a black box. At the bottom, the 'Next >' button is also highlighted with a black box.

Create Servlet
Enter servlet deployment descriptor specific information.

Name: LoginServlet
Description:

Initialization parameters:

Name	Value	Description
------	-------	-------------

URL mappings:

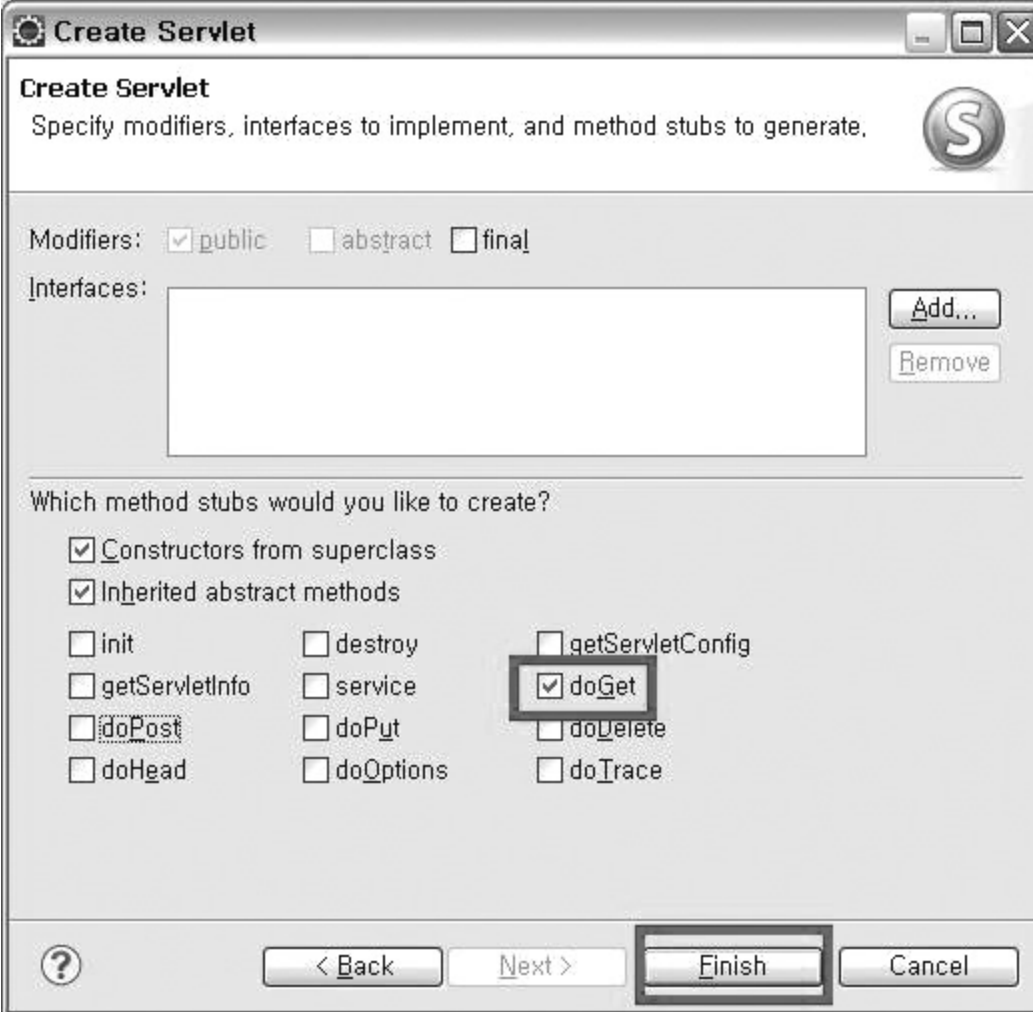
/login

< Back **Next >** Finish Cancel



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

- 05. 본 예제의 요청 방식이 GET 방식이므로 Which method stubs would you like to create? 부분의 체크 박스에서 doGet 메소드만 체크한 후 [Finish] 버튼을 클릭한다.



Create Servlet

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

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces: Add... Remove

Which method stubs would you like to create?

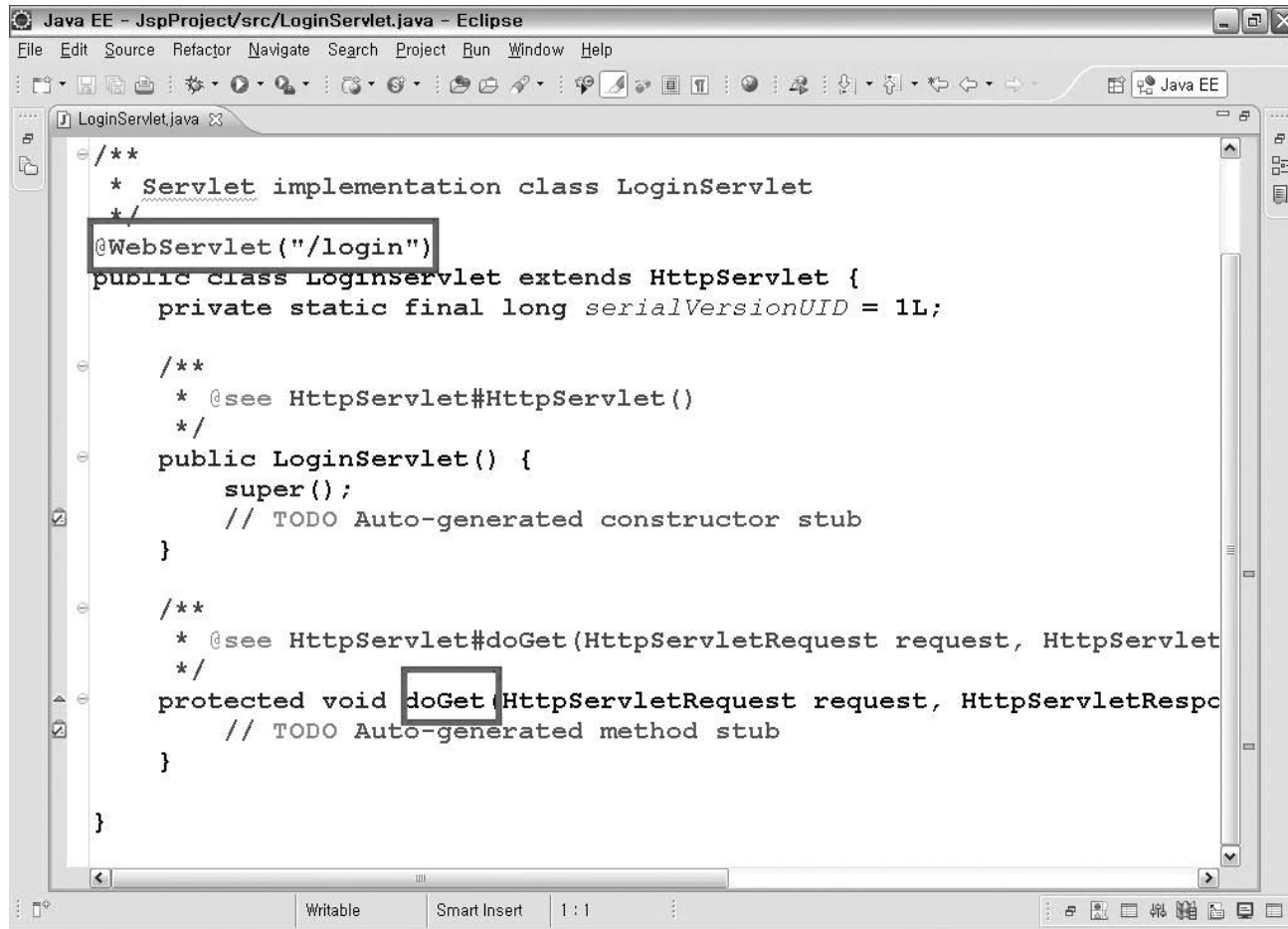
<input checked="" type="checkbox"/> Constructors from superclass		
<input checked="" type="checkbox"/> Inherited abstract methods		
<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet
<input type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

? < Back Next > **Finish** Cancel



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

- 06. 다음과 같은 템플릿 페이지 생성



```
Java EE - JspProject/src/LoginServlet.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

LoginServlet.java
/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public LoginServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        // TODO Auto-generated method stub
    }
}
```



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

- **jsp2.2**는 서블릿 설정이 web.xml보다는 어노테이션* 기반을 우선 제공하고 있으므로 @WebServlet(“/login”)코드가 자동으로 생성된 것이다.
- 이 부분은 `http://localhost:8088/JspTest/login`으로 요청이 전송되어 오면 해당 서블릿 클래스에서 요청을 처리하겠다는 의미이다.

어노테이션(Annotation)

Java 5.0 부터 지원되는 기술로, 기존 설정 파일(web.xml 등)에서 제공하는 설정 내용들을 설정 파일에서 설정하지 않아도 해당 소스 내에 설정할 수 있는 방법을 제공함으로써 설정 파일의 크기를 줄이거나 설정 파일 자체를 없앨 수 있는 역할을 하는 기능이다.



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

• 07. LoginServlet.java 소스 코드 작성

```
1  import java.io.IOException;
2  import java.io.PrintWriter;
3
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 LoginServlet
12  */
13 @WebServlet("/login")
14 public class LoginServlet extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
```



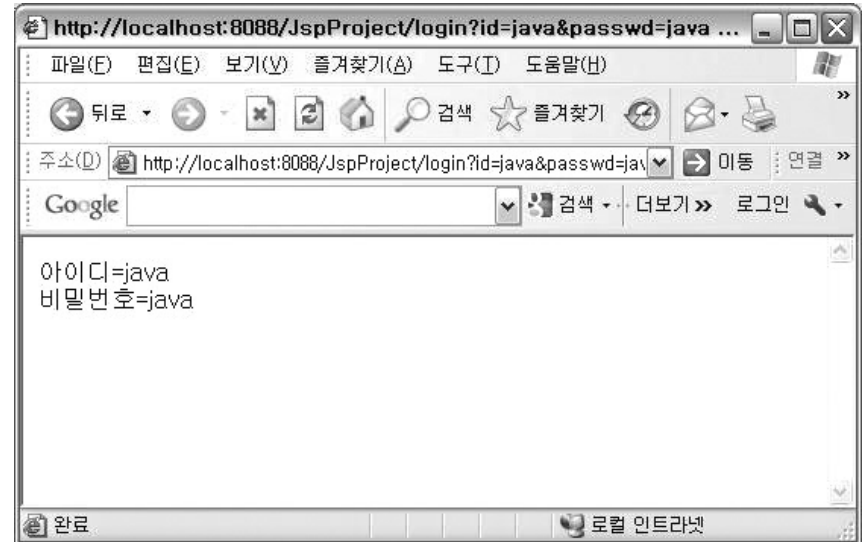
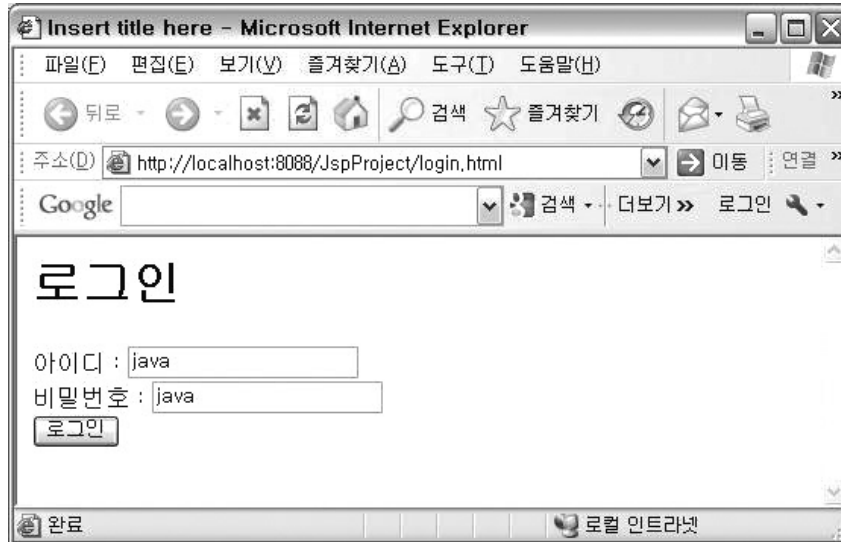
서블릿을 이용한 클라이언트에서 전송되는 요청 처리

```
17  /**
18  * @see HttpServlet#HttpServlet()
19  */
20  public LoginServlet() {
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
29  ServletException, IOException {
30      // TODO Auto-generated method stub
31      String id = request.getParameter("id");
32      String passwd = request.getParameter("passwd");
33      response.setContentType("text/html;charset=euc-kr");
34      PrintWriter out = response.getWriter();
35      out.println("아이디="+id + "<br>");
36      out.println("비밀번호="+passwd + "<br>");
37  }
38
39 }
```



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

■ 결과 확인하기



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

□ POST 방식으로 요청이 전송되어 올 경우

■ 클라이언트 페이지 코딩 - memReg.html

```
1  <body>
2  <h1>회원가입</h1>
3  <form action="memReg" method="post">
4      회원명 : <input type="text" name="name"><br>
5      주소 : <input type="text" name="addr"><br>
6      전화번호 : <input type="text" name="tel"><br>
7      취미 : <input type="text" name="hobby"><br>
8      <input type="submit" value="회원가입"/>
9  </form>
10 </body>
```



서블릿을 이용한 클라이언트에서 전송되는 요청 처리

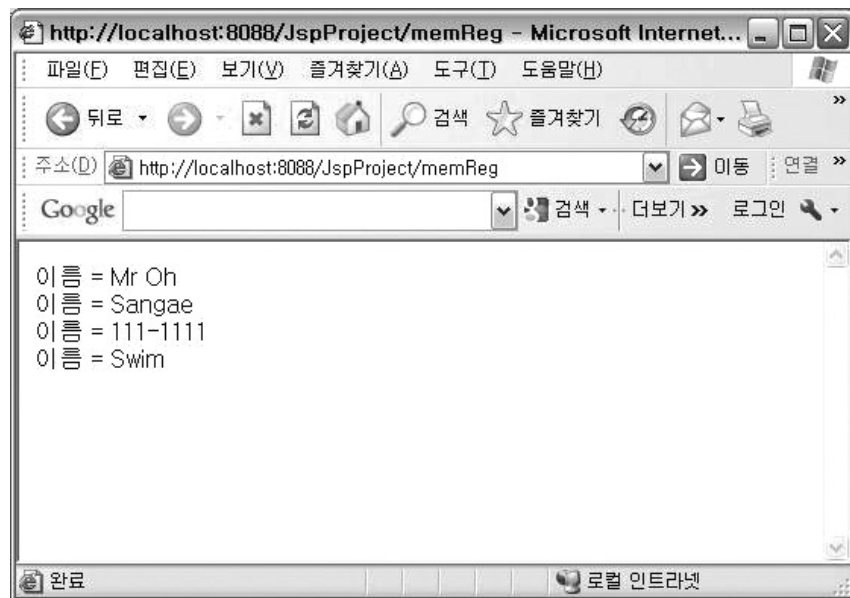
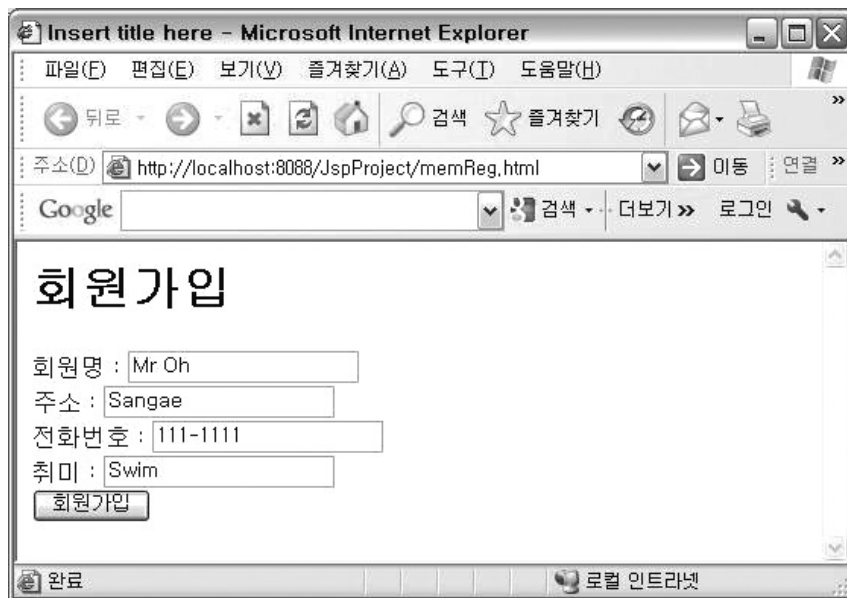
■ 서블릿 페이지 코딩 - MemRegServlet.java

```
1  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
2  ServletException, IOException {
3      // TODO Auto-generated method stub
4      response.setContentType("text/html;charset=euc-kr");
5      PrintWriter out = response.getWriter();
6      String name = request.getParameter("name");
7      String addr = request.getParameter("addr");
8      String tel = request.getParameter("tel");
9      String hobby = request.getParameter("hobby");
10     out.println("이름 = "+name + "<br>");
11     out.println("이름 = "+addr + "<br>");
12     out.println("이름 = "+tel + "<br>");
13     out.println("이름 = "+hobby + "<br>");
14 }
```



서블릿을 이용한 클라이언트에서 전송되는 요청 처리


■ 결과 확인



서블릿에서 한글 처리하기

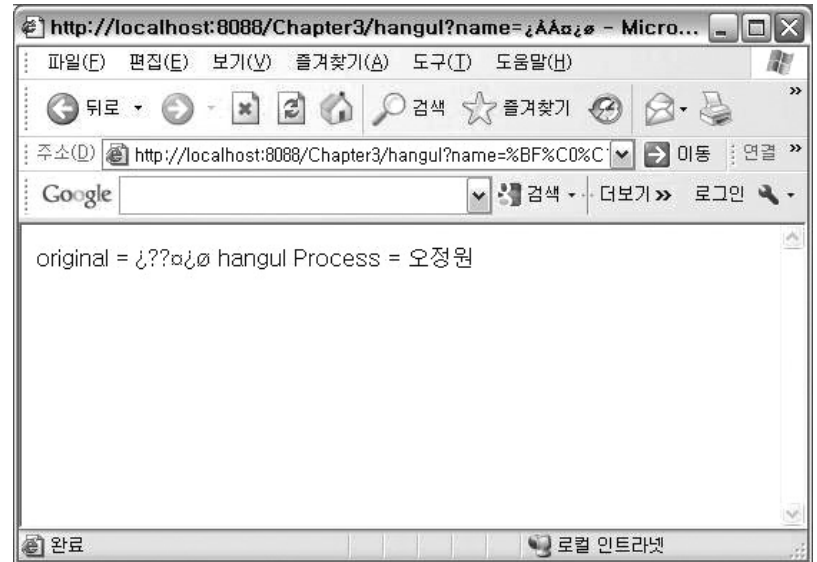
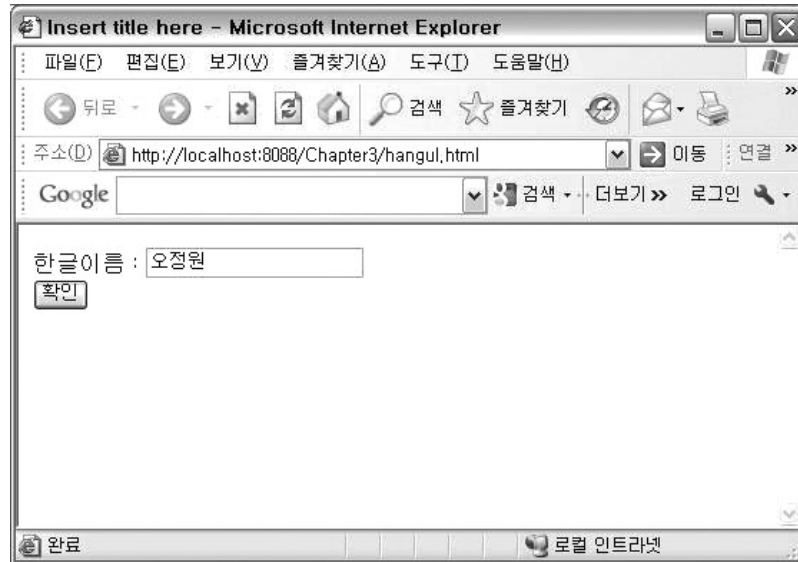
□ GET 방식으로 요청이 전송되어 올 경우

- 한글 처리용 클라이언트 페이지 작성
 - `<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">`

hangul.html		 Chapter3\Wex3\Whangul.html
1	<code><html></code>	
2	<code><head></code>	
3	<code><meta http-equiv="Content-Type" content="text/html; charset=EUC-KR"></code>	
4	<code><title>Insert title here</title></code>	
5	<code></head></code>	
6	<code><body></code>	
7	<code><form action="hangul" method="get"></code>	
8	<code> 한글이름 : <input type="text" name="name"/>
</code>	
9	<code> <input type="submit" value="확인"/></code>	
10	<code></form></code>	
11	<code></body></code>	
12	<code></html></code>	

서블릿에서 한글 처리하기

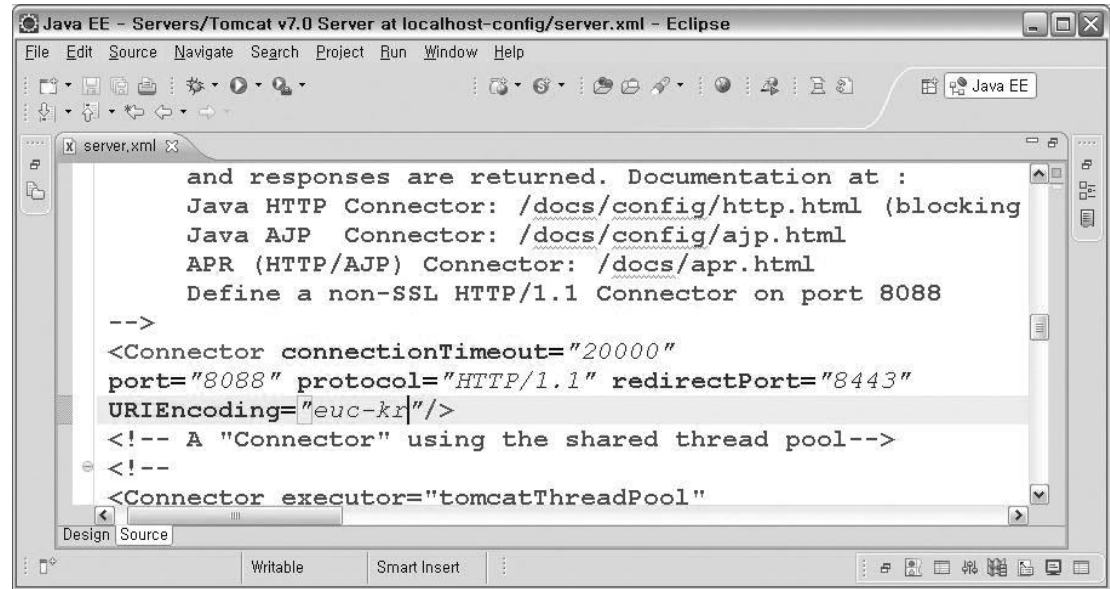
- 실행 결과



- **server.xml**을 수정해서 **GET** 방식으로 전송되어 오는 파라미터들의
캐릭터 셋을 한 번에 수정하는 방법
 - **Connector** 태그에 **URIEncoding= "euc-kr"** 을 추가한 후 서버 재시작

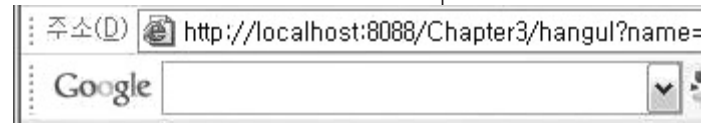


서블릿에서 한글 처리하기



■ 서블릿 코드 수정

```
1 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
2 ServletException, IOException {
3     // TODO Auto-generated method stub
4     String name = request.getParameter("name");
5     String korName = name;
6
7     response.setContentType("text/html;charset=euc-kr");
8     PrintWriter out = response.getWriter();
9     out.println("original = " + name);
10    out.println("hangul Process = " + korName);
11 }
```

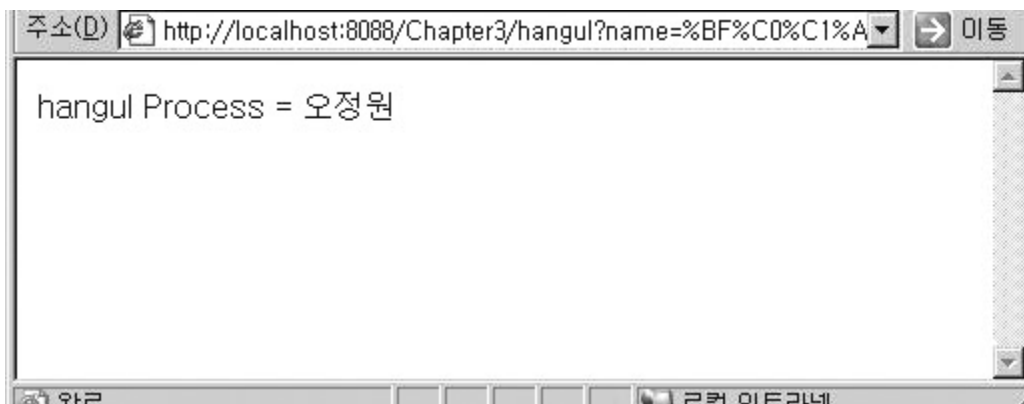


original = 오정원 hangul Process = 오정원

서블릿에서 한글 처리하기

- **server.xml connector 태그의 useBodyEncodingForURI= “true” 속성을 이용하여 한글 처리하는 방법**
 - **Connector 태그에 useBodyEncodingForURI= “true” 속성을 추가한 후 서버 재시작**

```
1  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
2  ServletException, IOException {
3      // TODO Auto-generated method stub
4      request.setCharacterEncoding("euc-kr");
5      String name = request.getParameter("name");
6      response.setContentType("text/html;charset=euc-kr");
7      PrintWriter out = response.getWriter();
8      out.println("hangul Process = " + name);
9  }
```



서블릿에서 한글 처리하기

□ POST 방식으로 요청이 전송되어 올 경우

- Html 페이지의 코딩에서 다음 폼 태그의 method 속성을 POST로 변경

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
4  <title>Insert title here</title>
5  </head>
6  <body>
7  <form action="hangul" method="POST">
8      한글이름 : <input type="text" name="name"/><br>
9      <input type="submit" value="확인"/>
10 </form>
11 </body>
12 </html>
```



서블릿에서 한글 처리하기

- 서블릿 페이지에서는 `doPost`에서 `request.setCharacterEncoding("euc-kr")`을 처리

```
1  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
2  ServletException, IOException {
3      // TODO Auto-generated method stub
4      request.setCharacterEncoding("euc-kr");
5      String name = request.getParameter("name");
6      response.setContentType("text/html;charset=euc-kr");
7      PrintWriter out = response.getWriter();
8      out.println("hangul Process = " + name);
9  }
```



하나의 파라미터 이름으로 여러 개의 파라미터 값 처리하기

□ HttpServletRequest 인터페이스 이용


- 하나의 파라미터 값이 전송되는 경우
 - `String getParameter(String paramName)` 메소드를 사용
- 하나의 파라미터 이름으로 여러 개의 값이 전송되어 올 경우
 - `String[] getParameterValues(String paramName)` 메소드를 사용

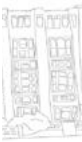


하나의 파라미터 이름으로 여러 개의 파라미터 값 처리하기

■ 예제

• 클라이언트 페이지 - dog.html

dog.html		 Chapter3Wex4Wdog.html
1	<html>	
2	<head>	
3	<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">	
4	<title>Insert title here</title>	
5	</head>	
6	<body>	
7	<h1>당신이 좋아하는 강아지를 선택 하세요</h1>	
8	<form action="choiceDog" method="post">	
9	<input type="checkbox" name="dog" value="pu.jpg"/>푸들	
10	<input type="checkbox" name="dog" value="jin.jpg"/>진돗개	
11	<input type="checkbox" name="dog" value="pung.jpg"/>풍산개	
12	<input type="checkbox" name="dog" value="sap.jpg"/>삽살개	
13	<input type="submit" value="선택"/>	
14	</form>	
15	</body>	
16	</html>	



하나의 파라미터 이름으로 여러 개의 파라미터 값 처리하기

- 서블릿 페이지 - ChoiceDogServlet.java

ChoiceDogServlet.java

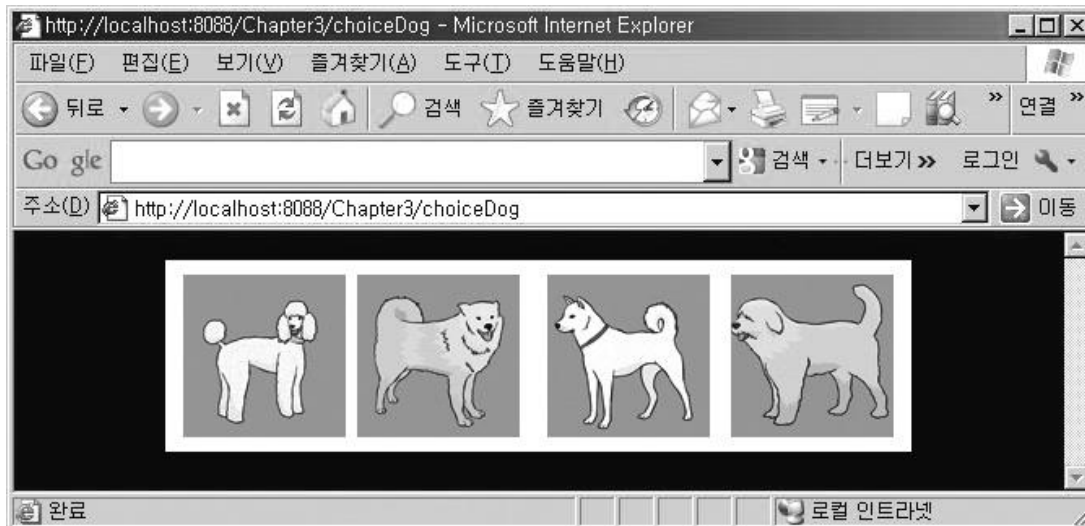
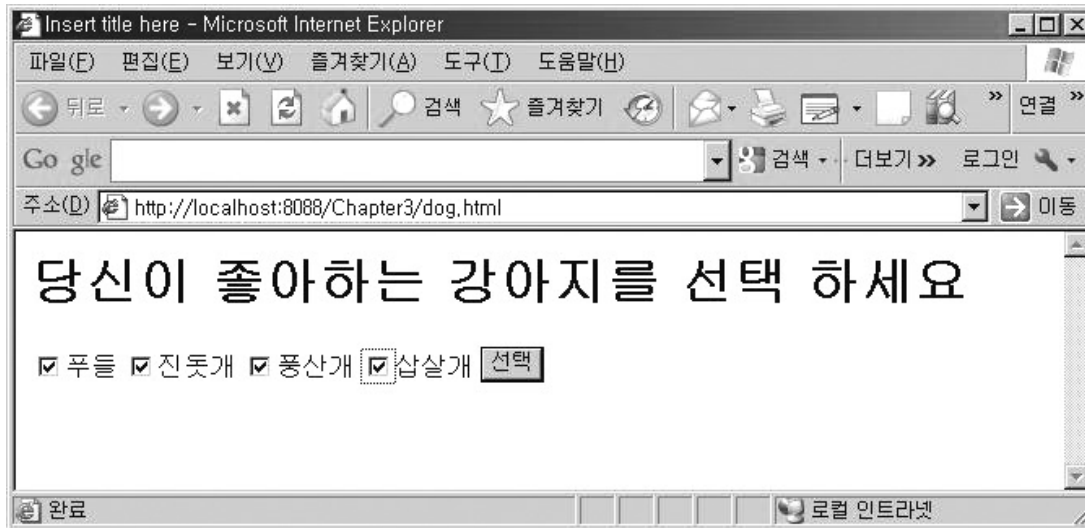


Chapter3Web4ChoiceDogServlet.java

```
1  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
2  ServletException, IOException {
3      // TODO Auto-generated method stub
4      response.setContentType("text/html;charset=euc-kr");
5      PrintWriter out = response.getWriter();
6      String[] dog = request.getParameterValues("dog");
7      out.println("<html>");
8      out.println("<head>");
9      out.println("</head>");
10     out.println("<body bgcolor='black'>");
11     out.println("<table align='center' bgcolor='yellow'>");
12     out.println("<tr>");
13     for(int i=0;i<dog.length;i++){
14         out.println("<td>");
15         out.println("<img src='"+dog[i]+'"/>");
16         out.println("</td>");
17     }
18     out.println("</tr>");
19     out.println("</table>");
20     out.println("</body>");
21     out.println("</html>");
22 }
```

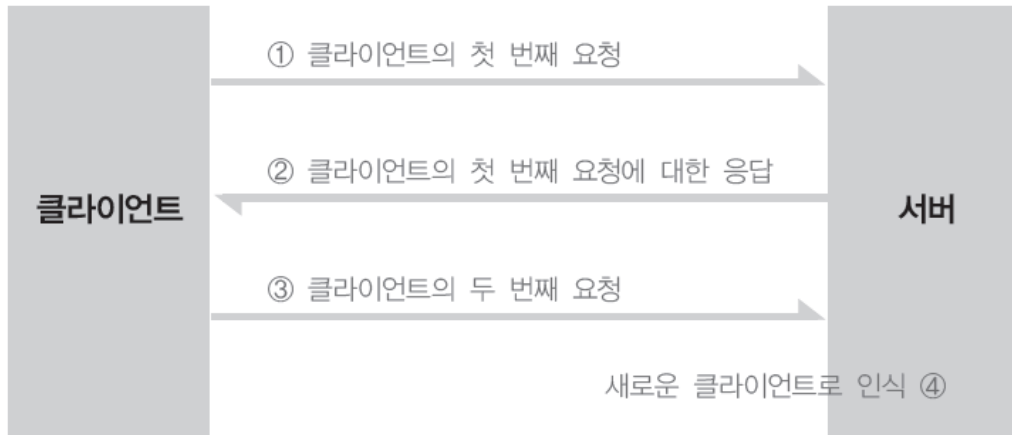
하나의 파라미터 이름으로 여러 개의 파라미터 값 처리하기

- 결과 확인



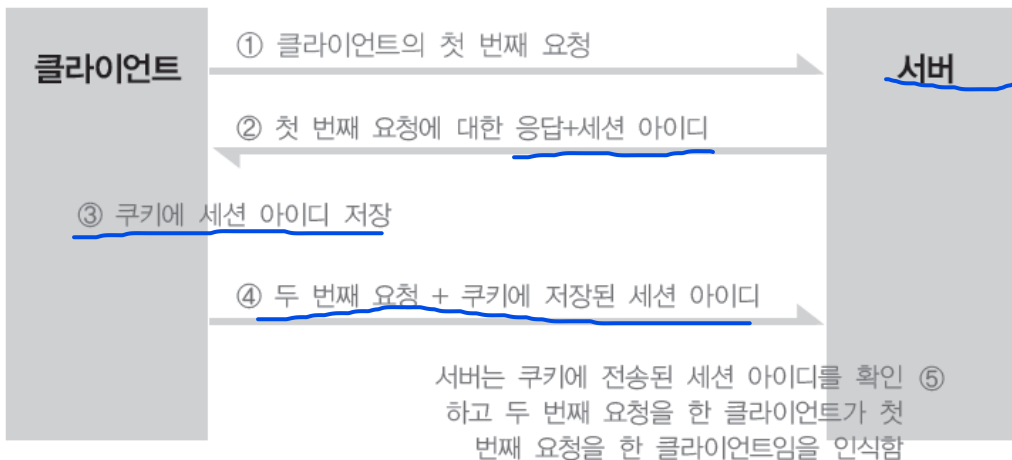
서블릿에서 세션 살펴보기

□ 세션의 개념



Stateless
상태유지X

[그림 3-3] HTTP 프로토콜의 상태를 유지하지 않는 속성



세션키보따리
← 세션 아이디

[그림 3-4] 세션 기능이 적용된 HTTP 프로토콜 요청 처리



서블릿에서 세션 살펴보기

□ 간단한 세션 예제

■ HttpSession 인터페이스 사용

SetNameServlet.java



Chapter3\Wex5\SetNameServlet.java

```
1 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
2 ServletException, IOException {
3     // TODO Auto-generated method stub
4     HttpSession session = request.getSession();
5     session.setAttribute("name","오정원");
6     response.setContentType("text/html;charset=euc-kr");
7     PrintWriter out = response.getWriter();
8     out.println("<h1>이름저장</h1>");
9 }
```

GetNameServlet.java의 doGet



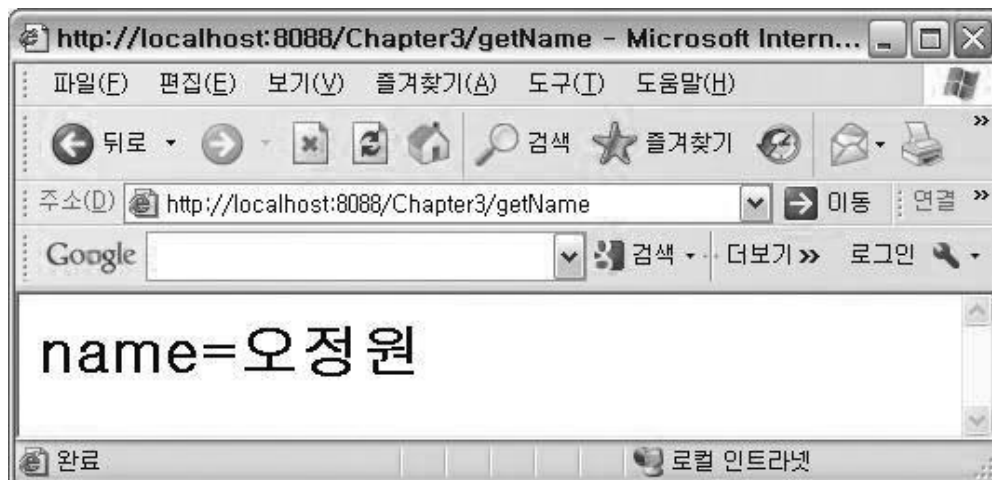
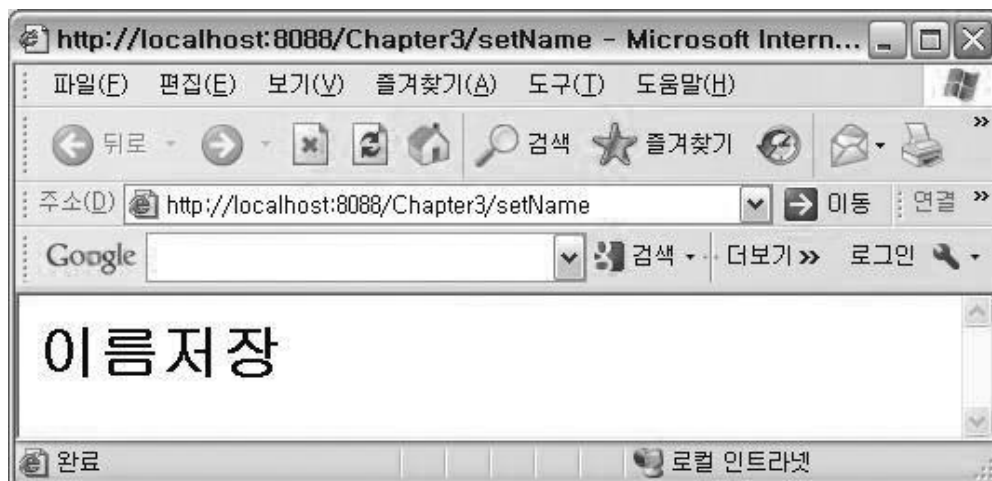
Chapter3\Wex5\GetNameServlet.java

```
1 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
2 ServletException, IOException {
3     // TODO Auto-generated method stub
4     HttpSession session = request.getSession();
5     String name=(String)session.getAttribute("name");
6     response.setContentType("text/html;charset=euc-kr");
7     PrintWriter out = response.getWriter();
8     out.println("<h1>name="+name+"</h1>");
9 }
```



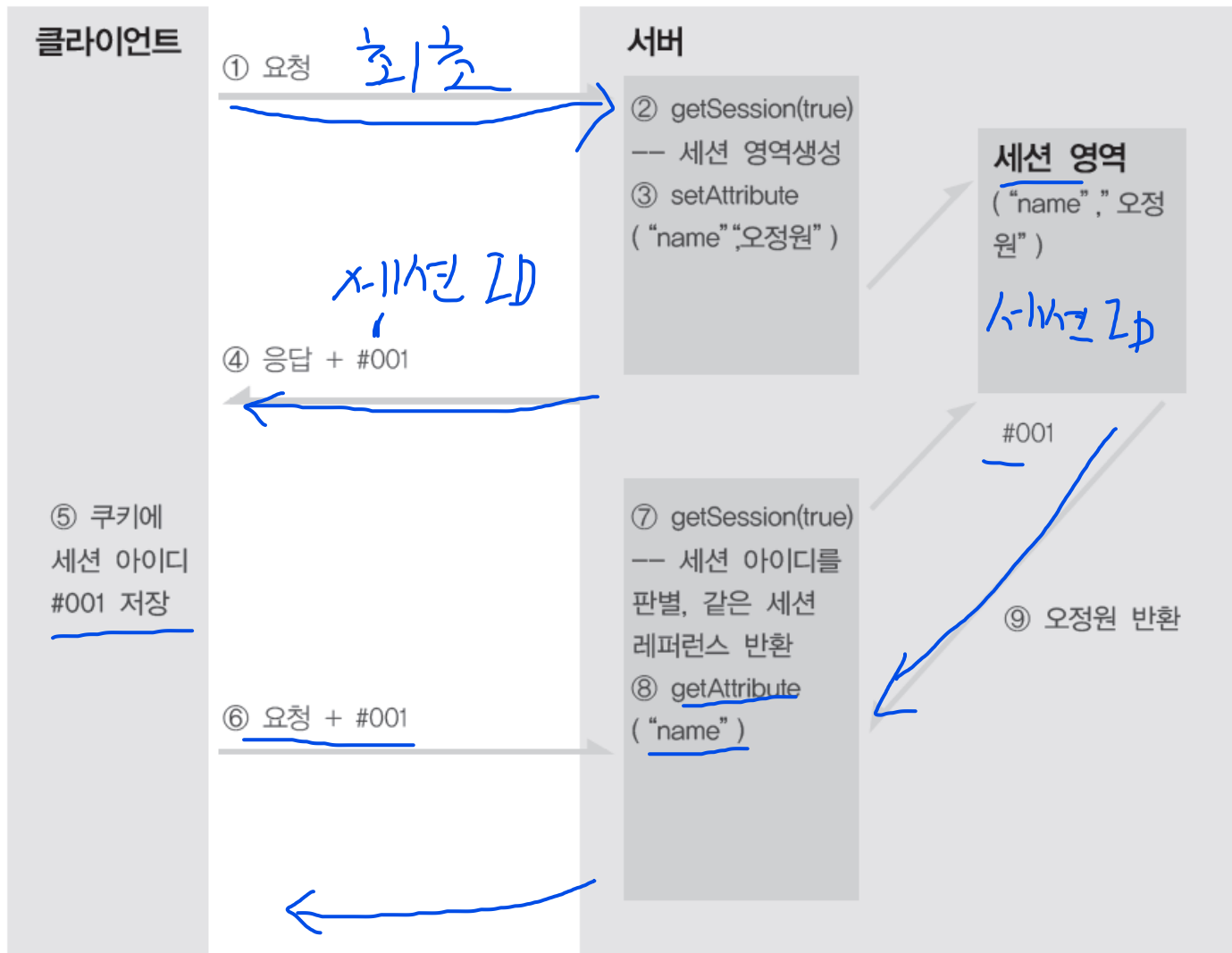
서블릿에서 세션 살펴보기

■ 결과 확인



서블릿에서 세션 살펴보기

동작 과정




[그림 3-5] 세션 객체를 생성하여 자신의 세션 객체에 속성 저장하고 사용하기



서블릿에서 세션 살펴보기

□ 세션 기능을 이용해 로그인 프로그램 구현해 보기

- 메인 페이지를 index.html로 작성한다. 프레임을 좌측과 우측 프레임으로 나눈다.

index.html		 Chapter3\Wex6\index.html
1	<frameset cols="30%,*">	
2	<frame src="menu.jsp" name="leftFrame"/>	
3	<frame src="login.html" name="rightFrame"/>	
4	</frameset>	
5	<body>	
6	</body>	



서블릿에서 세션 살펴보기

- login.jsp를 작성한다.

login.jsp



Chapter3Wex6Wlogin.jsp

```
1 <form action="login" method="post">
2     아이디 : <input type="text" name="id"/>
3     비밀번호 : <input type="password" name="passwd"/><br>
4     <input type="submit" value="로그인"/>
5 </form>
```



서블릿에서 세션 살펴보기

■ menu.jsp를 작성한다

menu.jsp




Chapter3\Wex6\Wmenu.jsp

```
1  <%
2      String id = (String)session.getAttribute("id");
3  %>
4  <body>
5  <%
6      if(id == null){
7  %>
8      <a href="login.jsp" target="rightFrame"/>로그인</a>
9  <%
10     }
11     else{
12 %>
13     <%=id %> 님 환영합니다.
14 <%
15     }
16 %>
```

서블릿에서 세션 살펴보기

- **loginSuccess.jsp**를 작성한다. 이 페이지는 서블릿에서 로그인 성공 후 포워딩되는 페이지이다.

loginSuccess.jsp		 Chapter3\Wex6\loginSuccess.jsp
1	<head>	
2	<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">	
3	<title>Insert title here</title>	
4	<script>	
5	top.leftFrame.location.href="menu.jsp";	
6	</script>	
7	</head>	
8	<body>	
9	로그인 성공	
10	</body>	



서블릿에서 세션 살펴보기

- **LoginServlet.java**를 작성한다.

LoginServlet.java



Chapter3Wex6WLoginServlet.java

```
1  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
2  ServletException, IOException {
3      // TODO Auto-generated method stub
4      request.setCharacterEncoding("euc-kr");
5      response.setContentType("text/html;charset=euc-kr");
6      PrintWriter out = response.getWriter();
7
8
9      String id=request.getParameter("id");
10     String passwd = request.getParameter("passwd");
11     if(id.equals("java")&& passwd.equals("java")){
12         HttpSession session = request.getSession();
13         session.setAttribute("id", id);
14         RequestDispatcher dispatcher =
15             request.getRequestDispatcher("loginSuccess.jsp");
16         dispatcher.forward(request, response);
17     }
```

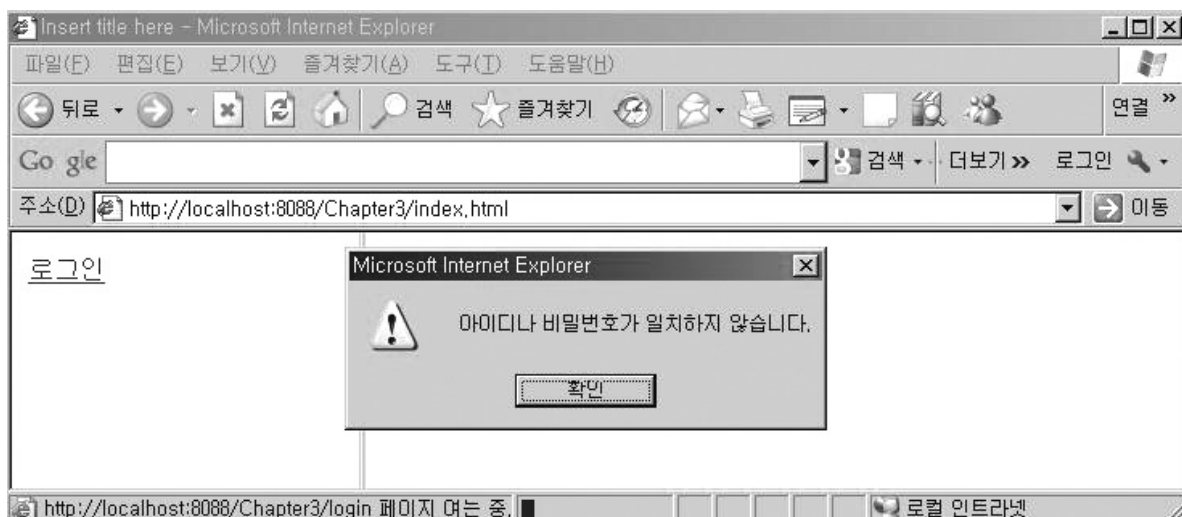
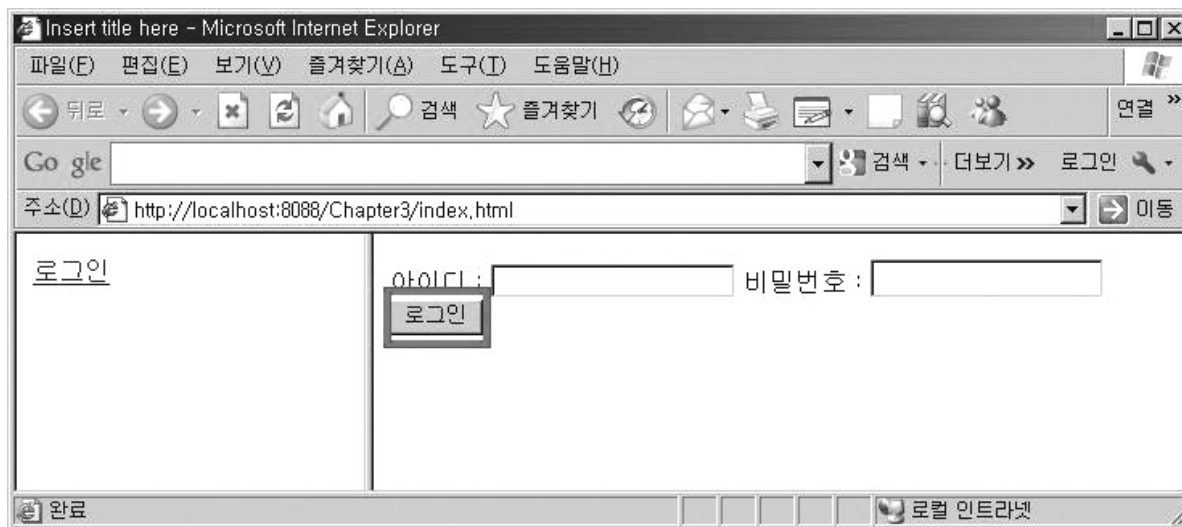
서블릿에서 세션 살펴보기

```
18         else{
19             out.println("<script>");
20             out.println("alert('아이디나 비밀번호가 일치하지 않습니다.');"");
21             out.println("history.back()");
22             out.println("</script>");
23         }
24     }
```

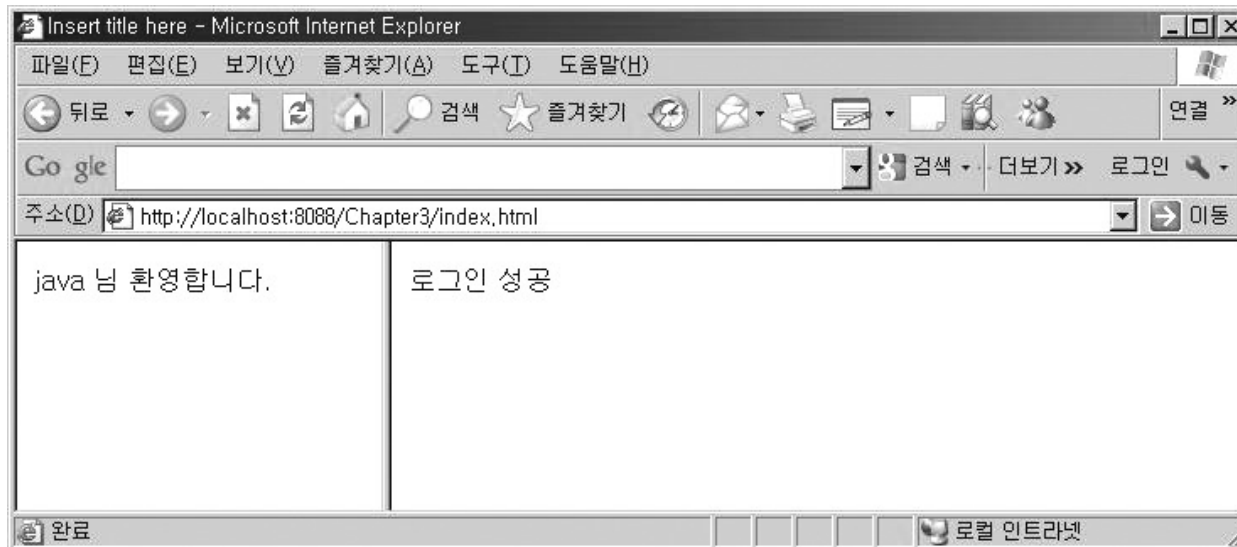
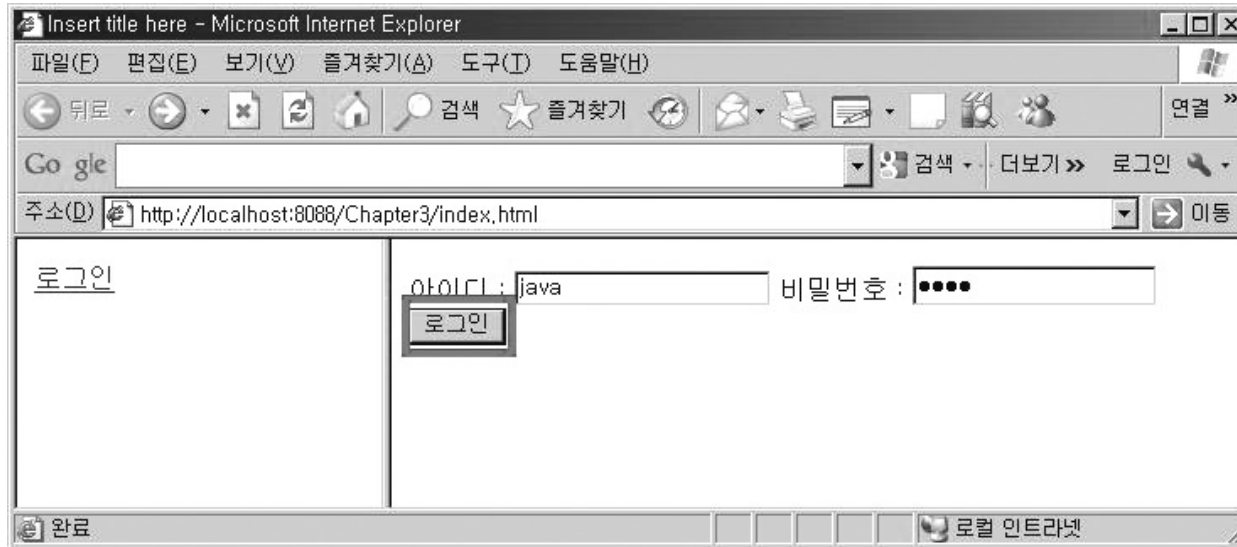


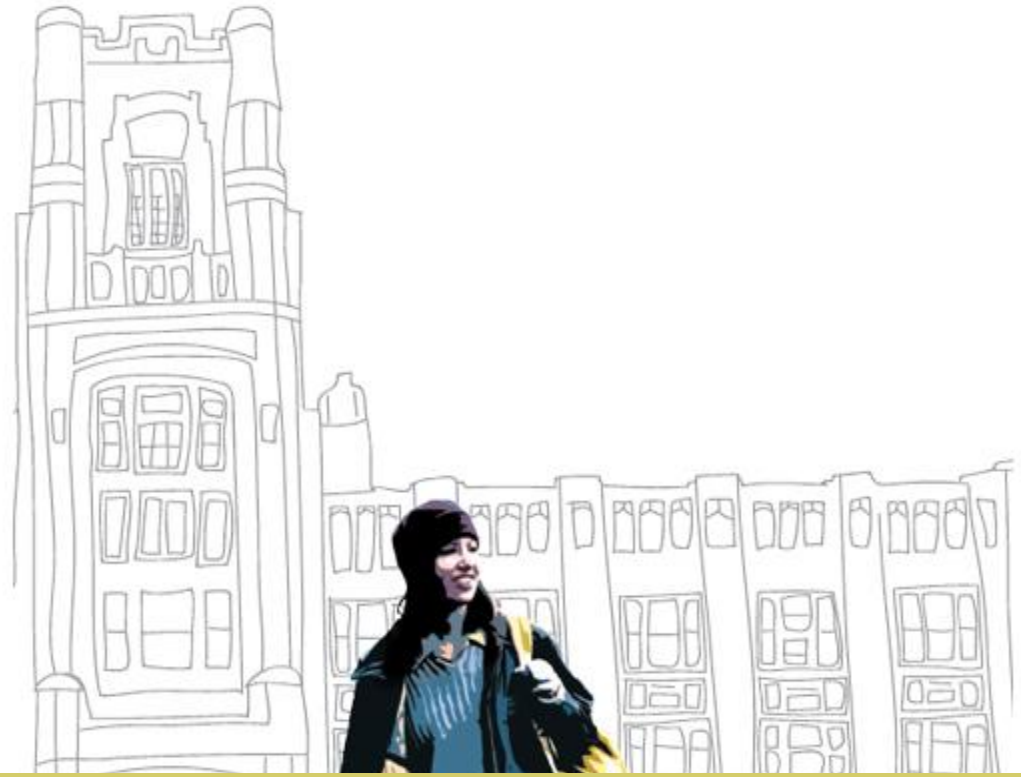
서블릿에서 세션 살펴보기

■ 결과 확인 - index.html로 접속



서블릿에서 세션 살펴보기





Thank You

