

5장 서블릿 기초

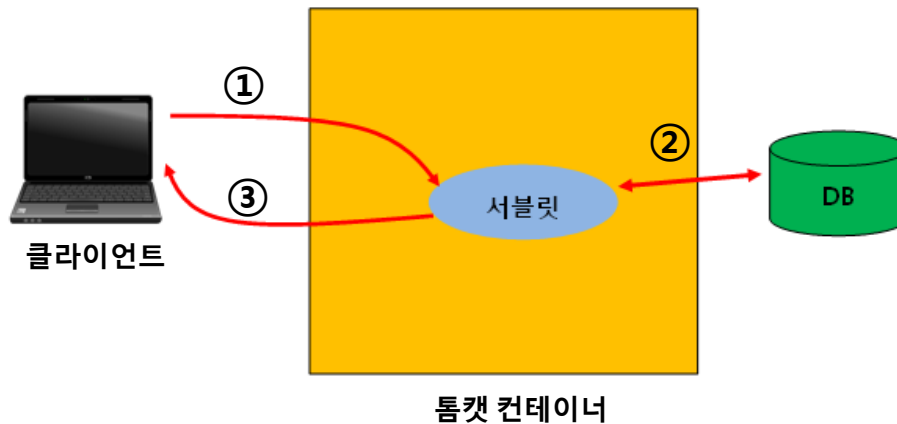
1. 서블릿 세 가지 기본 기능
2. `<form>` 태그 이용해 서블릿에 요청하기
3. 서블릿에서 클라이언트의 요청 얻는 방법
4. 서블릿의 응답 처리 방법
5. 웹 브라우저에서 서블릿으로 데이터 전송하기
6. GET 방식과 POST 방식 요청 동시에 처리하기
7. 자바스크립트로 서블릿에 요청하기
8. 서블릿을 이용한 여러 가지 실습 예제

1. 서블릿의 세 가지 기본 기능

- 1.1 서블릿 기본 기능 수행 과정

➤ 초기의 웹 프로그래밍에선 서블릿을 이용해서 브라우저의 요청을 처리해서 서비스를 제공했음

- 서블릿의 세 가지 기본 기능



- ① 클라이언트로부터 요청을 얻음
- ② 데이터베이스 연동과 같은 비즈니스 로직을 처리함
- ③ 처리된 결과를 클라이언트에 응답

1. 서블릿의 세 가지 기본 기능

1.2 서블릿 요청과 응답 수행 API 기능

- 요청과 관련된 API: javax.servlet.http.HttpServletRequest 클래스
- 응답과 관련된 API: javax.servlet.http.HttpServletResponse 클래스

- 요청과 응답 관련 API 사용 예

```
public class FirstServlet extends HttpServlet {  
    @Override  
    public void init() throws ServletException {  
        System.out.println("init 메서드 호출");  
    }  
  
    @Override  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
    {  
        System.out.println("doGet 메서드 호출");  
    }  
  
    @Override  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
    {  
        System.out.println("doGet 메서드 호출");  
    }  
  
    @Override  
    public void destroy() {  
        System.out.println("destroy 메서드 호출");  
    }  
}
```

1. 서블릿의 세 가지 기본 기능

HttpServletRequest의 여러가지 메서드

반환형	메서드 이름	기능
boolean	Authenticate (HttpServletRequest response)	현재 요청한 사용자가 ServletContext 객체에 대한 인증을 하기 위한 컨테이너 로그인 메커니즘을 사용
String	changeSessionId()	현재 요청과 연관된 현재 세션의 id를 변경하여 새 세션 id를 반환
String	getContextPath()	요청한 컨텍스트를 가리키는 URI를 반환
Cookie[]	getCookies()	클라이언트가 현재의 요청과 함께 보낸 쿠키 객체들에 대한 배열을 반환.
String	getHeader(String name)	특정 요청에 대한 헤더 정보를 문자열로 반환
Enumeration <String>	getHeaderNames	현재의 요청에 포함된 헤더의 name 속성을 enumeration으로 반환
String	getMethod()	현재 요청이 GET, POST 또는 PUT 방식 중 어떤 HTTP 요청인지를 반환
String	getRequestURI()	요청한 URL의 컨텍스트 이름과 파일 경로까지 반환
String	getServletPath()	요청한 URL에서 서블릿이나 JSP 이름을 반환
HttpSession	getSession()	현재의 요청과 연관된 세션을 반환합니다. 만약 세션이 없으면 새로 만들어서 반환
String	getPathInfo()	클라이언트가 요청 시 보낸 URL과 관련된 추가 경로 정보를 반환

1. 서블릿의 세 가지 기본 기능

HttpServletResponse의 여러가지 메서드

반환형	메서드 이름	기능
void	addCookie (Cookie cookie)	응답에 쿠키를 추가
void	addHeader(String name, String value)	name과 value를 헤더에 추가
String	encodeURL(String url)	클라이언트가 쿠키를 지원하지 않을 때 세션 id를 포함한 특정 URL을 인코딩
Collection <String>	getHeaderNames()	현재 응답의 헤더에 포함된 name을 얻어옴.
void	sendRedirect (String location)	클라이언트에게 리다이렉트(redirect) 응답을 보낸 후 특정 URL로 다시 요청

2. <form> 태그를 이용해 서블릿에 요청

2.1 <form> 태그로 서블릿에 요청하는 과정



2. <form> 태그를 이용해 서블릿에 요청

2.2 <form> 태그의 여러 가지 속성

아이디 :

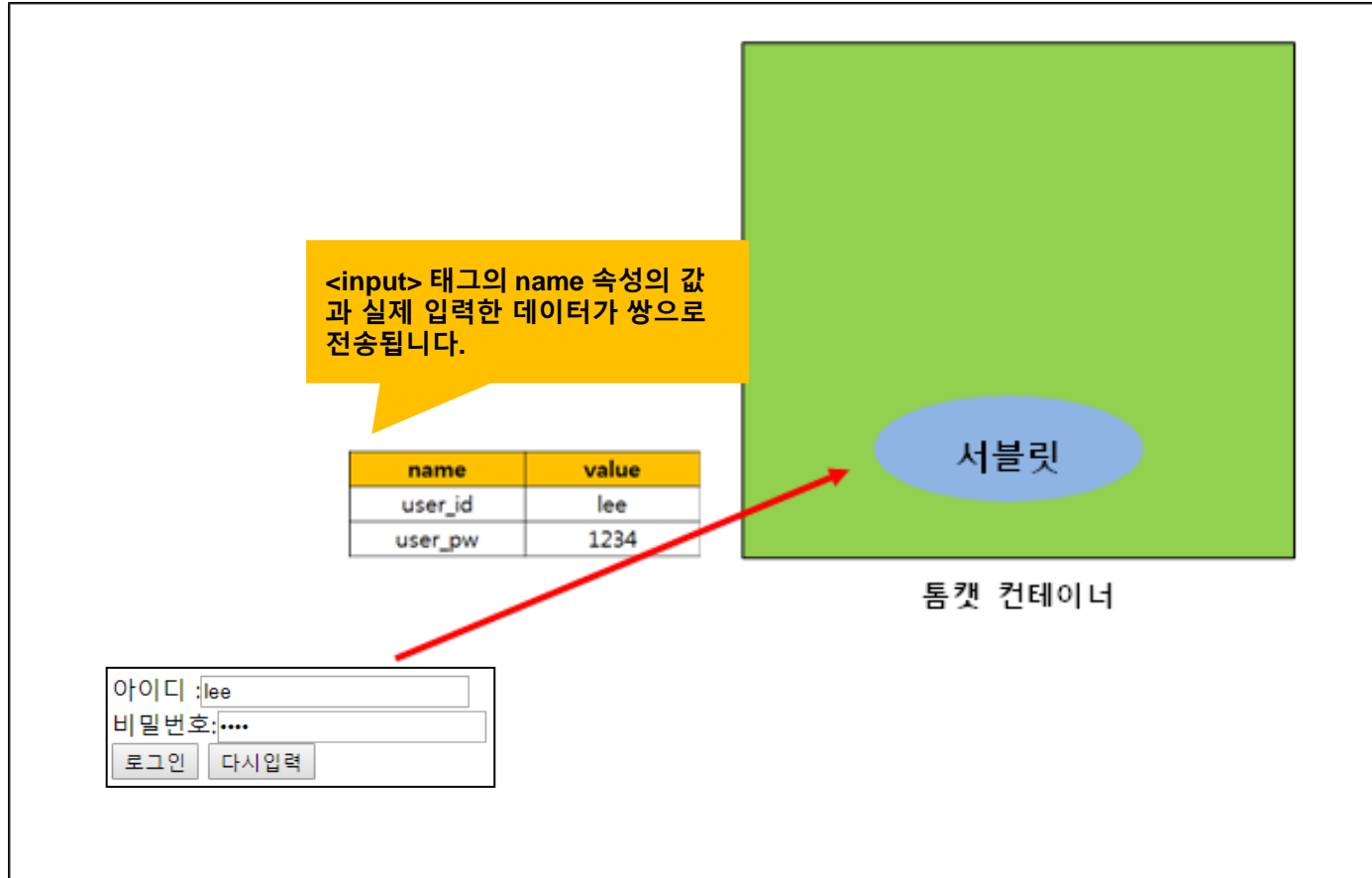
비밀번호:



```
<form name="frmLogin" method="get" action="login" encType="UTF-8">  
  아이디 :<input type="text" name="user_id"><br>  
  비밀번호:<input type="password" name="user_pw" ><br>  
  <input type="submit" value="로그인"> <input type="reset" value="다시입력">  
</form>
```

2. <form> 태그를 이용해 서블릿에 요청

➤ 로그인 버튼 클릭 시 <form> 태그의 action 속성에 지정한 JSP나 서블릿으로 name/value로 전송



2. <form> 태그를 이용해 서블릿에 요청

<form> 태그와 관련된 여러 가지 속성

속성	기능
name	<ul style="list-style-type: none">• <form> 태그의 이름을 지정.• 여러 개의 form이 존재할 경우 구분• 자바스크립트에서 <form> 태그에 접근할 때 자주 사용
method	<ul style="list-style-type: none">• <form> 태그 안에서 데이터를 전송할 때 전송 방법을 지정.• GET 또는 POST로 지정(아무것도 지정하지 않으면 GET).
action	<ul style="list-style-type: none">• <form> 태그에서 데이터를 전송할 서블릿이나 JSP를 지정• 서블릿으로 전송할 때는 매핑 이름을 사용
encType	<ul style="list-style-type: none">• <form> 태그에서 전송할 데이터의 encoding 타입을 지정• 파일을 업로드할 때는 multipart/form-data로 지정

3. 서블릿에서 클라이언트의 요청 얻는 방법

➤ HttpServletRequest 클래스의 여러가지 메서드를 이용해서 전송된 데이터를 얻음

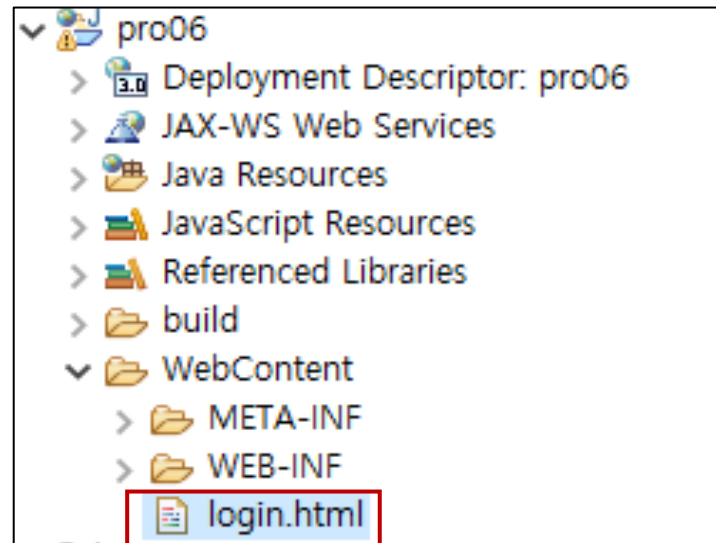
<form> 태그로 전송된 데이터를 받아 오는 메서드

메서드	기능
String getParameter(String name)	name의 값을 알고 있을 때 그리고 name에 대한 전송된 값을 받아오는 데 사용
String[] getParameterValues(String name)	같은 name에 대해 여러 개의 값을 얻을 때 사용
Enumeration getParameterNames()	name 값을 모를 때 사용

3. 서블릿에서 클라이언트의 요청 얻는 방법

3.1 HttpServletRequest로 요청 처리 실습

1. pro06이라는 새 프로젝트를 생성
2. WebContent 폴더 하위에 사용자 정보를 입력 받을 login.html을 생성



3. 서블릿에서 클라이언트의 요청 얻는 방법

3. 다음과 같이 login.html 파일 작성. 로그인창에서 ID와 비밀번호를 입력 받은 후 서블릿으로 전송

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>로그인 창</title>
</head>
<body>
  <form name="frmLogin" method="get" action="login" encType="UTF-8">
    아이디 :<input type="text" name="user_id"><br>
    비밀번호:<input type="password" name="user_pw"><br>
    <input type="submit" value="로그인"> <input type="reset" value="다시입력">
  </form>
</body>
</html>
```

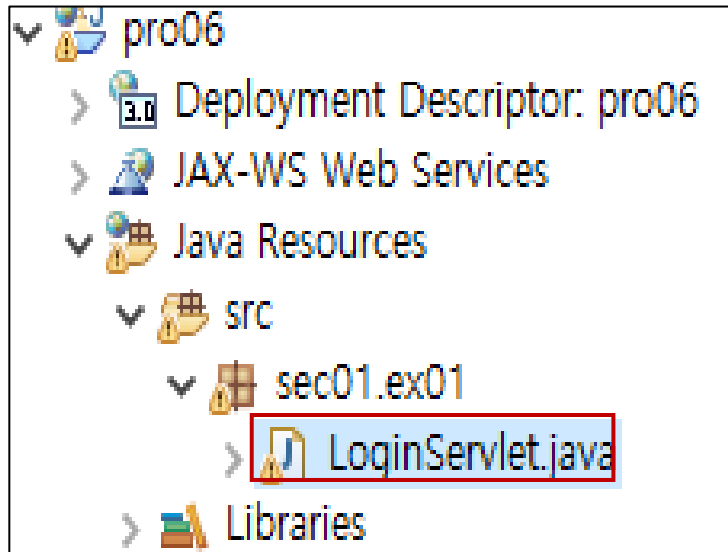
텍스트 박스에 입력된 ID를 user_id로 전송합니다.

입력된 데이터를 서블릿 매핑 이름이 login인 서블릿으로 전송합니다.

텍스트 박스에 입력된 비밀번호를 user_pw로 전송합니다.

3. 서블릿에서 클라이언트의 요청 얻는 방법

4. sec01.ex01 패키지를 만들고 요청을 받을 서블릿인 LoginServlet 클래스를 생성



3. 서블릿에서 클라이언트의 요청 얻는 방법

5. 다음과 같이 LoginServlet.java 코드를 작성. HttpServletRequest 클래스의 getParameter() 메서드로 전송된 ID와 비밀번호를 받아 옴.

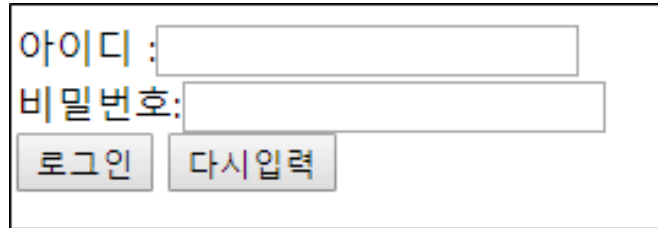
```
...
@WebServlet("/login") * 서블릿의 매핑 이름이 login입니다.
public class LoginServlet extends HttpServlet
{
    public void init() throws ServletException
    {
        System.out.println("init 메서드 호출");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
    {
        request.setCharacterEncoding("utf-8"); * 전송된 데이터를 UTF-8로 인코딩합니다.
        String user_id = request.getParameter("user_id"); * getParameter()를 이용해 <input>
        String user_pw = request.getParameter("user_pw"); * 태그의 name 속성 값으로 전송된
        System.out.println("아이디:" + user_id); * value를 받아옵니다.
        System.out.println("비밀번호:" + user_pw);
    }

    public void destroy()
    {
        System.out.println("destroy 메서드 호출");
    }
}
```

3. 서블릿에서 클라이언트의 요청 얻는 방법

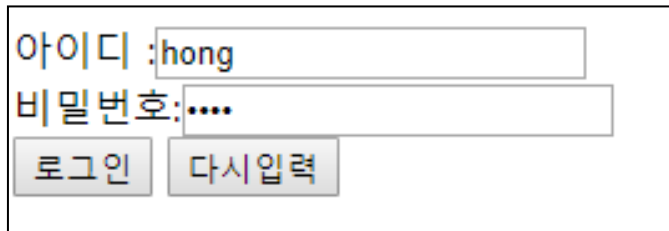
6. pro06 프로젝트를 톰캣에 등록하여 실행한 후 브라우저에서 <http://localhost:8090/pro06/login.html>을 요청



아이디 :

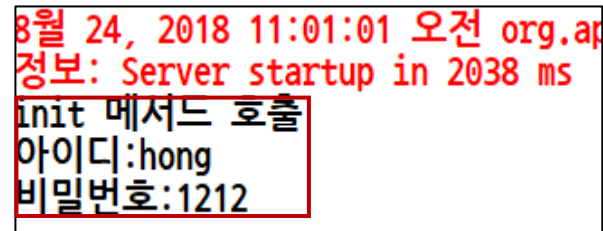
비밀번호:

7. 텍스트 박스에 ID와 비밀번호를 입력한 후 로그인을 클릭하면 서블릿이 ID와 비밀번호를 이클립스 콘솔에 출력



아이디 : hong

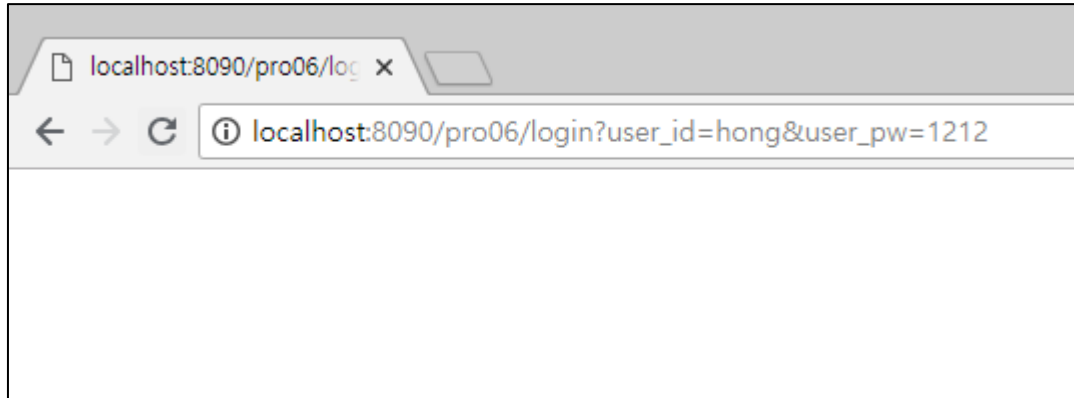
비밀번호:



```
8월 24, 2018 11:01:01 오전 org.ap
정보: Server startup in 2038 ms
init 메서드 호출
아이디:hong
비밀번호:1212
```

3. 서블릿에서 클라이언트의 요청 얻는 방법

단, 서블릿이 처리한 후의 응답 기능은 아직 구현하지 않았으므로 웹 브라우저에는 아무것도 출력되지 않습니다.

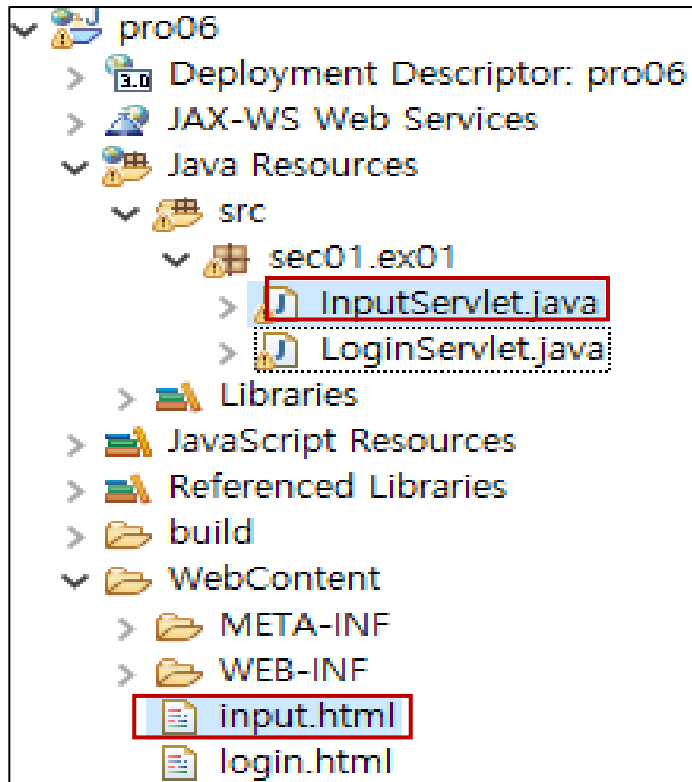


3. 서블릿에서 클라이언트의 요청 얻는 방법

3.2 여러 개의 값을 전송할 때의 요청 처리

➤ 한번에 여러 개의 값이 전송되는 경우 `getParameterValues()` 메서드를 이용

1. 다음과 같이 `input.html`을 추가하고 `InputServlet` 클래스를 새로 작성



3 서블릿에서 클라이언트의 요청 얻는 방법

2. input.html을 다음과 같이 작성 <input> 타입이 여러 개일 때는 체크박스(Checkbox)를 사용해서 값을 설정. 체크박스의 name 속성 값은 모두 subject이므로 서블릿으로 전송할 때 배열로 전송.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>여러 가지 input 타입 표시 창</title>
</head>
<body>
  <form name="frmInput" method="get" action="input">
    아이디 :<input type="text" name="user_id"><br>
    비밀번호:<input type="password" name="user_pw"><br>
    <input type="checkbox" name="subject" value="java" checked>자바
    <input type="checkbox" name="subject" value="C언어">C언어
    <input type="checkbox" name="subject" value="JSP">JSP
    <input type="checkbox" name="subject" value="안드로이드">안드로이드
    <br><br>
    <input type="submit" value="전송">
    <input type="reset" value="초기화">
  </form>
</body>
</html>
```

name 속성이 모두 subject로 같습니다.

전송을 클릭하면 매핑 이름이 action에 설정한 input 서블릿으로 전송됩니다.

3. 서블릿에서 클라이언트의 요청 얻는 방법

3. InputServlet 클래스를 다음과 같이 작성. `getParameterValues()`를 이용해 `input.html`에서 체크박스의 `name`인 `subject`로 전송된 값들을 받아 와서 문자열 배열에 저장

```
...
@WebServlet("/input")
public class InputServlet extends HttpServlet
{
    public void init() throws ServletException
    {
        System.out.println("init 메서드 호출");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        request.setCharacterEncoding("utf-8");
        String user_id = request.getParameter("user_id");
        String user_pw = request.getParameter("user_pw");
        System.out.println("아이디:" + user_id);
        System.out.println("비밀번호:" + user_pw);
        String[] subject = request.getParameterValues("subject");
        for (String str : subject)
        {
            System.out.println("선택한 과목:" + str);
        }
    }

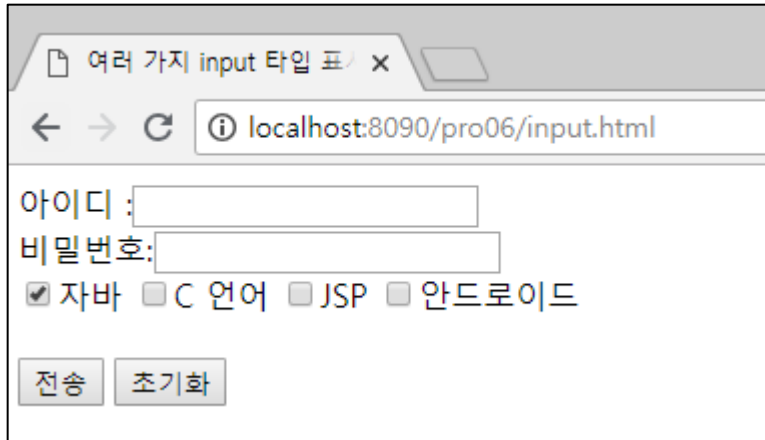
    public void destroy()
    {
        System.out.println("destroy 메서드 호출");
    }
}
```

한 개씩 전송된 값은
`getParameter()`를 이
용합니다.

하나의 `name`으로 여러 값을 전송하는
경우 `getParameterValues()`를 이용해
배열 형태로 반환됩니다.

3. 서버릿에서 클라이언트의 요청 얻는 방법

4. 브라우저에서 <http://localhost:8090/pro06/input.html>로 요청



The screenshot shows a web browser window with the address bar displaying `localhost:8090/pro06/input.html`. The page content includes a form with the following elements:

- A tab labeled "여러 가지 input 타입 표" with a close button.
- Navigation icons: back, forward, and refresh.
- Input fields for "아이디 :" (ID) and "비밀번호:" (Password).
- Radio buttons for selecting a language: ☒ 자바 (Java), ☐ C 언어 (C Language), ☐ JSP, and ☐ 안드로이드 (Android).
- Buttons for "전송" (Submit) and "초기화" (Reset).

3. 서블릿에서 클라이언트의 요청 얻는 방법

5. 체크박스에서 여러 개의 값에 체크한 후 전송을 클릭하면 이클립스 콘솔에 해당 과목명이 출력

여러 가지 input 타입 표 x

localhost:8090/pro06/input.html

아이디 : hong

비밀번호:

☒ 자바 ☒ C 언어 ☒ JSP ☐ 안드로이드

전송 초기화

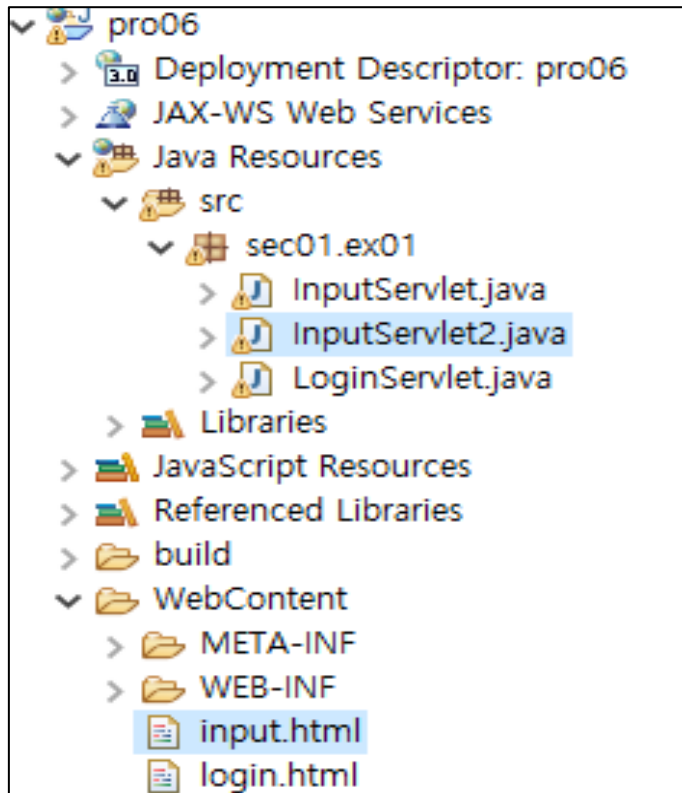
정보: Server startup in 1970 ms
init 메서드 호출
아이디:hong
비밀번호:1212
선택한 과목:java
선택한 과목:C 언어
선택한 과목:JSP

3. 서블릿에서 클라이언트의 요청 얻는 방법

3.3 getParameterNames() 메서드를 이용한 요청 처리

- 전송되는 데이터가 많은 경우 name의 값을 일일이 기억할 필요없이 getParameterNames() 메서드를 이용해서 name을 얻음

1. sec01.ex01 패키지에 InputServlet2 클래스를 생성



3. 블릿에서 클라이언트의 요청 얻는 방법

2. 그리고 3.2절에서 이용했던 input.html을 다음과 같이 수정

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>...</head>
```

```
<body>
```

```
<form name="frmInput" method="get" action="input2">
```

———— 매핑 이름을 input2로 수정합니다.

(중략)

3. 서블릿에서 클라이언트의 요청 얻는 방법

3. InputServlet2 클래스를 다음과 같이 작성. 전송되는 데이터가 많은 경우에는 `getParameterNames()`를 이용해 name 속성만 따로 구할 수 있다.

```
...
@WebServlet("/input2")
public class InputServlet2 extends HttpServlet
{
    public void init() throws ServletException
    {
        System.out.println("init 메서드 호출");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        request.setCharacterEncoding("utf-8");
        Enumeration enu = request.getParameterNames();
        while (enu.hasMoreElements())
        {
            String name = (String) enu.nextElement();
            String[] values = request.getParameterValues(name);
            for (String value : values)
            {
                System.out.println("name=" + name + ",value=" + value);
            }
        }
    }

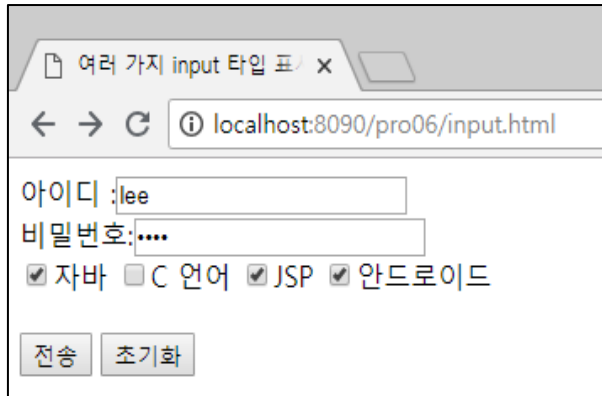
    public void destroy() {
        System.out.println("destroy 메서드 호출");
    }
}
```

전송되어 온 name 속성
들만 Enumeration 타입
으로 받아옵니다

각 name을 하나씩 가
져와 대응해서 전송되
어 온 값을 출력합니다.

3. 서버릿에서 클라이언트의 요청 얻는 방법

4. 브라우저에서 `http://localhost:8090/pro06/input.html`로 요청하고 값을 입력한 후 전송을 클릭
`getParameterNames()`를 이용해 전송된 name과 값이 모두 출력되는 것을 확인할 수 있다.



A screenshot of a web browser window. The address bar shows `localhost:8090/pro06/input.html`. The page contains a form with the following elements:

- A label "아이디" followed by a text input field containing the value "lee".
- A label "비밀번호:" followed by a text input field containing four dots "....".
- Four checkboxes: "자바" (checked), "C 언어" (unchecked), "JSP" (checked), and "안드로이드" (checked).
- Two buttons at the bottom: "전송" (Submit) and "초기화" (Reset).

```
정보: Server startup in 1895 ms  
init 메서드 호출  
name=user_id,value=lee  
name=user_pw,value=1234  
name=subject,value=java  
name=subject,value=JSP  
name=subject,value=안드로이드
```

4. 서블릿의 응답 처리

서블릿의 응답 처리 방법

- ① doGet()이나 doPost() 메서드 안에서 처리함
- ② javax.servlet.http.HttpServletResponse 객체를 이용함
- ③ setContentType()을 이용해 클라이언트에게 전송할 데이터 종류(MIME-TYPE)를 지정함
- ④ 클라이언트(웹 브라우저)와 서블릿의 통신은 자바 I/O의 스트림을 이용함

4.1 MINE-TYPE

➤ 톰캣 컨테이너에 미리 지정해 놓은 데이터 종류로 서블릿에서 브라우저로 전송 시 설정해서 사용함

MIME-TYPE 지정 예

- HTML로 전송 시: text/html
- 일반 텍스트로 전송 시: text/plain
- XML 데이터로 전송 시: application/xml

4. 서블릿의 응답 처리

톰캣 컨테이너의 web.xml에 정의된 여러 가지 MIME-TYPE

```
4435     <mime-mapping>
4436         <extension>xenc</extension>
4437         <mime-type>application/xenc+xml</mime-type>
4438     </mime-mapping>
4439     <mime-mapping>
4440         <extension>xer</extension>
4441         <mime-type>application/patch-ops-error+xml</mime-type>
4442     </mime-mapping>
4443     <mime-mapping>
4444         <extension>xfdf</extension>
4445         <mime-type>application/vnd.adobe.xfdf</mime-type>
4446     </mime-mapping>
4447     <mime-mapping>
4448         <extension>xfdl</extension>
4449         <mime-type>application/vnd.xfdl</mime-type>
4450     </mime-mapping>
4451     <mime-mapping>
4452         <extension>xht</extension>
4453         <mime-type>application/xhtml+xml</mime-type>
4454     </mime-mapping>
4455     <mime-mapping>
4456         <extension>xhtml</extension>
4457         <mime-type>application/xhtml+xml</mime-type>
4458     </mime-mapping>
```

4. 서블릿의 응답 처리

4.2 HttpServletResponse를 이용한 서블릿 응답 실습

setContentType()를 이용해 MIME-TYPE 지정



데이터를 출력할 PrintWriter 객체 생성



출력 데이터를 HTML형식으로 만듦



PrintWriter의 print()나 println()을 이용해 데이터 출력

4. 서블릿의 응답 처리

1. login.html을 다음과 같이 수정 로그인창에서 ID와 비밀번호를 입력한 후 login2 서블릿으로 전송.

```
<!DOCTYPE html>
<html>
<head>..</head>
<body>
  <form name="frmLogin" method="get" action="login2" encType="UTF-8">
    아이디:<input type="text" name="user_id"><br>
    비밀번호:<input type="password" name="user_pw"><br>
    <input type="submit" value="로그인"> <input type="reset" value="다시 입력">
  </form>
</body>
</html>
```

매핑 이름을 login2로 수정합니다.

4. 서블릿의 응답 처리

2. sec02.ex01 패키지에 LoginServlet2 클래스를 추가하고 다음과 같이 작성. 브라우저에서 전달받은 ID와 비밀번호를 HTML 태그로 만든 후 다시 브라우저로 응답함.

```
package sec02.ex01;

...

@WebServlet("/login2")
public class LoginServlet2 extends HttpServlet
{
    public void init() throws ServletException
    {
        System.out.println("init 메서드 호출");
    }
}
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
```

응답은 HttpServletResponse
객체를 이용합니다.

```
{
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html; charset=utf-8");
    PrintWriter out = response.getWriter();
    String id = request.getParameter("user_id");
    String pw = request.getParameter("user_pw");
```

웹 브라우저에서 전송된 데이터의
인코딩을 설정합니다.

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

setContentType()을 이용해
응답할 데이터 종류가 HTML
임을 설정합니다.

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

HttpServletResponse 객체의
getWriter()를 이용해 출력 스트림
PrintWriter 객체를 받아
옵니다.

```
    String id = request.getParameter("user_id");
    String pw = request.getParameter("user_pw");
```

```
    String data = "<html>";
    data += "<body>";
    data += "아이디 : " + id;
    data += "<br>";
    data += "패스워드 : " + pw;
    data += "</html>";
    data += "</body>";
```

브라우저로 출력할 데이터를 문자열로
연결해서 HTML 태그로 만듭니다.

```
    out.print(data);
```

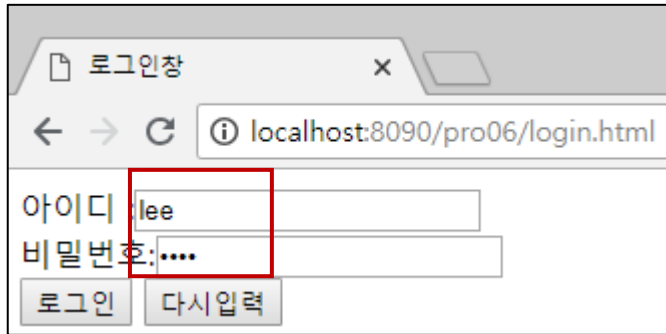
PrintWriter의 print()를 이용해 HTML 태그
문자열을 웹 브라우저로 출력합니다.

```
}
```

```
public void destroy() {
    System.out.println("destroy 메서드 호출");
}
}
```

4. 서블릿의 응답 처리

3. 브라우저에서 `http://localhost:8090/pro06/login.html`로 접속하여 ID와 비밀번호를 입력한 후 로그인을 클릭



로그인창 x

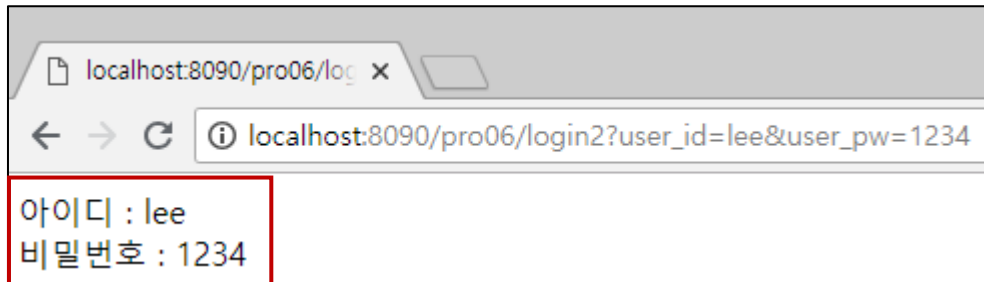
localhost:8090/pro06/login.html

아이디 lee

비밀번호:

로그인 다시입력

4. 그러면 서블릿이 ID와 비밀번호를 전달 받아 다시 브라우저로 출력



localhost:8090/pro06/log x

localhost:8090/pro06/login2?user_id=lee&user_pw=1234

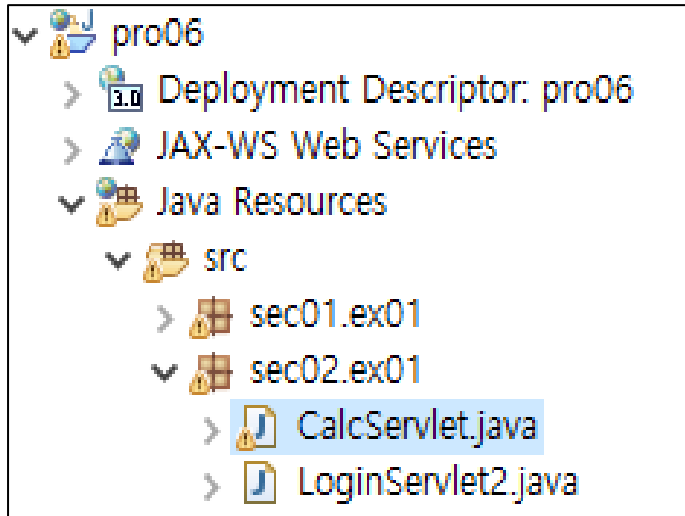
아이디 : lee

비밀번호 : 1234

4. 서블릿의 응답 처리

4.3 서블릿을 이용한 환율 계산기 예제 실습

1. sec02.ex01 패키지에 CalcServlet 클래스를 생성



4. 서블릿의 응답 처리

2. CalcServlet 클래스를 다음과 같이 작성 최초 매핑 이름인 /calc로 요청할 경우 command 값이 null이므로 환율 계산기 화면이 나타남. 계산기에서 값을 입력한 후 다시 요청할 경우 command 값이 calculate이므로 전달된 원화를 이용해 외화로 변환하여 결과를 출력

```
...
@WebServlet("/calc")
public class CalcServlet extends HttpServlet
{
    ...
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset=utf-8");
        PrintWriter pw = response.getWriter();
        String command = request.getParameter("command");
        String won = request.getParameter("won");
        String operator = request.getParameter("operator");
```



```
private static float USD_RATE = 1124.70F;
private static float JPY_RATE = 10.113F;
private static float CNY_RATE = 163.30F;
private static float GBP_RATE = 1444.35F;
private static float EUR_RATE = 1295.97F;
```

수행할 요청을 받아 옵니다.

변환할 원화를 받아 옵니다.

변환할 외화 종류를 받아 옵니다.

```

if (command != null && command.equals("calculate"))
{
    String result = calculate(Float.parseFloat(won), operator);
    pw.print("<html><font size=10>변환 결과</font><br>");
    pw.print("<html><font size=10>" + result + "</font><br>");
    pw.print("<a href='/pro06/calc'>환율 계산기</a>");
    return;
}

```

최초 요청 시 command
가 null이면 계산기 화면
을 출력하고, command
값이 calculate이면 계산
결과를 출력합니다.

```

pw.print("<html><title>환율 계산기</title>");
pw.print("<font size=5>환율 계산기</font><br>");
pw.print("<form name='frmCalc' method='get' action='/pro06/calc' /> ");
pw.print("원화: <input type='text' name='won' size=10 /> ");
pw.print("<select name='operator' >");
pw.print("<option value='dollar'>달러</option>");
pw.print("<option value='en'>엔화</option>");
pw.print("<option value='wian'>위안</option>");
pw.print("<option value='pound'>파운드</option>");
pw.print("<option value='euro'>유로</option>");
pw.print("</select>");
pw.print("<input type='hidden' name='command' value='calculate' />");
pw.println("<input type='submit' value='변환' />");
pw.println("</form>");
pw.print("</html>");

```

환율 정보 입력 후 다시 서블릿
calc로 요청합니다.

선택트 박스에서 선택된 값을
name으로 전송합니다.

<hidden> 태그를 이용해 계산
기에서 서블릿으로 수행할 요
청을 전달합니다.

```
    pw.close();  
}
```

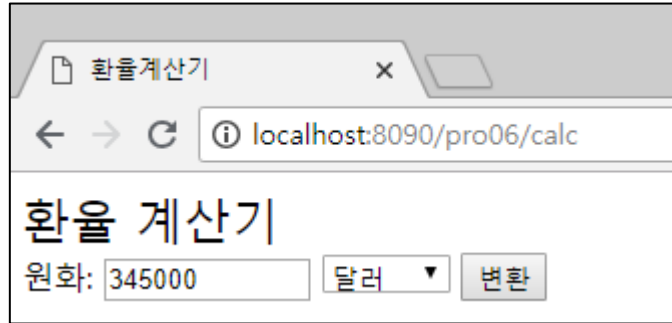
원화를 선택한 외화로
환산합니다.

```
private static String calculate(float won, String operator) {  
    String result = null;  
    if (operator.equals("dollar")) {  
        result = String.format("%.6f", won / USD_RATE);  
    } else if (operator.equals("en")) {  
        result = String.format("%.6f", won / JPY_RATE);  
    } else if (operator.equals("wian")) {  
        result = String.format("%.6f", won / CNY_RATE);  
    } else if (operator.equals("pound")) {  
        result = String.format("%.6f", won / GBP_RATE);  
    } else if (operator.equals("euro")) {  
        result = String.format("%.6f", won / EUR_RATE);  
    }  
    return result;  
}
```

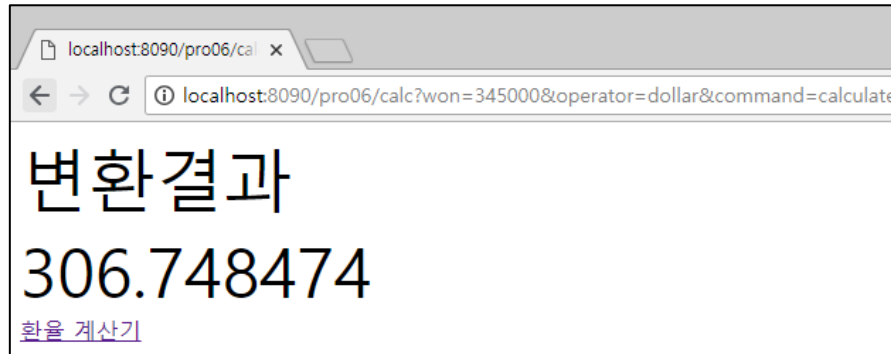
```
}
```

.4 서버릿의 응답 처리

3. 웹 브라우저에서 `http://localhost:8090/pro06/calc`로 요청한 후 원화에 값을 입력하고 변환을 클릭

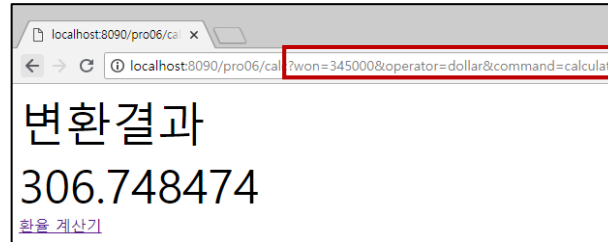


4. 값을 전송한 후 결과를 웹 브라우저에 출력합니다.



5. 웹 브라우저에서 서블릿으로 데이터 전송

5.1 GET/POST 전송 방식



GET 방식과 POST 방식 비교

GET 방식	POST 방식
<ul style="list-style-type: none">• 서블릿에 데이터를 전송할 때는 데이터가 URL 뒤에 name=value 형태로 전송• 여러 개의 데이터를 전송할 때는 '&'로 구분해서 전송.• 보안이 취약• 전송할 수 있는 데이터는 최대 255자임• 기본 전송 방식이고 사용이 쉬움.• 웹 브라우저에 직접 입력해서 전송할 수도 있음• 서블릿에서는 doGet()으로 전송된 데이터를 처리	<ul style="list-style-type: none">• 서블릿에 데이터를 전송할 때는 TCP/IP 프로토콜 데이터의 HEAD 영역에 숨겨진 채 전송• 보안에 유리• 전송 데이터 용량이 무제한• 전송 시 서블릿에서는 또다시 가져오는 작업을 해야 하므로 처리 속도가 GET 방식보다 느림• 서블릿에서는 doPost()를 이용해 데이터를 처리

5. 웹 브라우저에서 서블릿으로 데이터 전송

5.2 GET 방식으로 서블릿에 요청

➤ method= "get"은 서블릿에 GET방식으로 데이터를 전송하겠다는 의미

```
8 <form name= "frmInput" method= "get" action= "input">
9   아이디 : <input type= "text" name= "user_id"> <br>
10  비밀번호: <input type= "password" name= "user_pw"> <br>
11   <input type= "checkbox" name= "subject" value= "java" checked > 자바
12   <input type= "checkbox" name= "subject" value= "C 언어"> C 언어
```

➤ 서블릿에선 GET방식으로 전송된 데이터를 doGet() 메서드를 처리해야함.

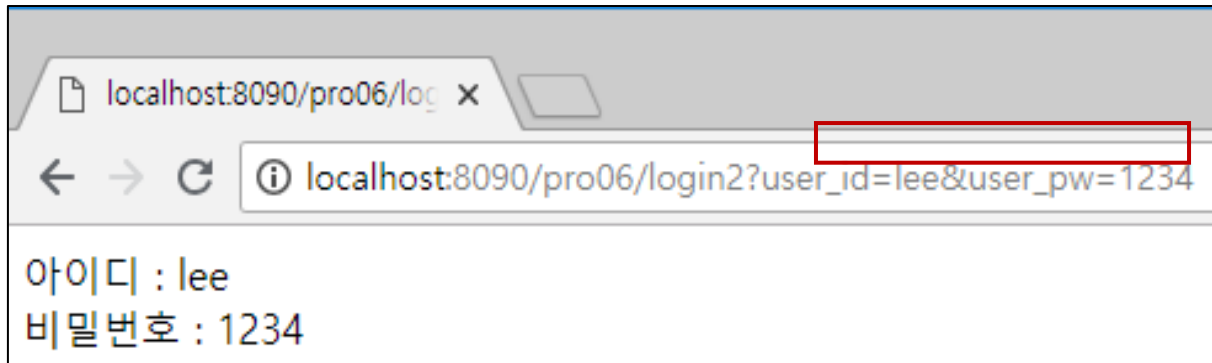
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();

    String id = request.getParameter("user_id");
    String pw = request.getParameter("user_pw");

    String data="<html>";
    data+="<body>";
    data+="아이디 : " + id ;
    data+="<br>";
    data+="패스워드 : " + pw;
    data+="</html>";
    data+="</body>";
    out.print(data);
}
```

5. 웹 브라우저에서 서블릿으로 데이터 전송

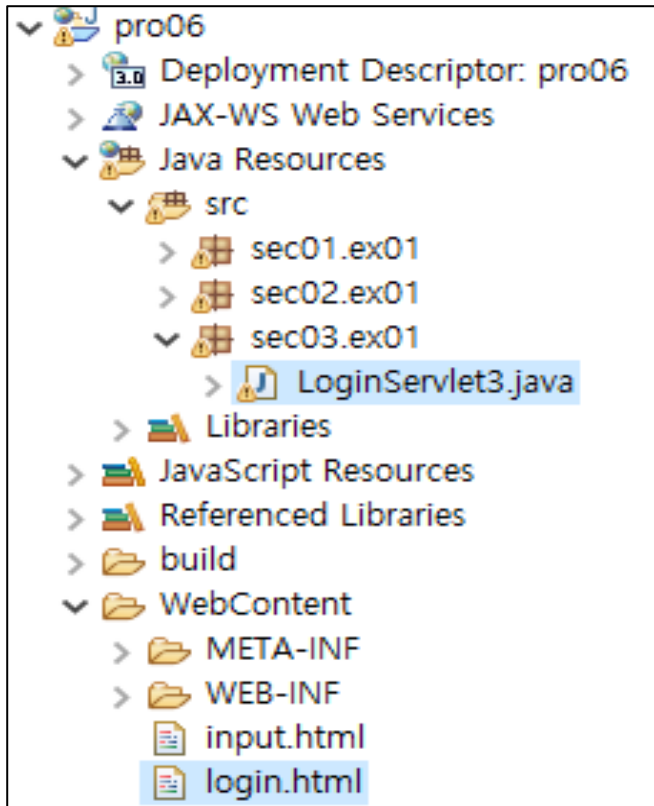
- 웹 브라우저의 주소창을 보면 `http://localhost:8090/pro06/login2?user_id=lee&user_pw=1234`처럼 URL 뒤에 'name=value' 쌍으로 붙어서 전송됨.



5. 웹 브라우저에서 서블릿으로 데이터 전송

5.3 POST 방식으로 서블릿에 요청

1. sec03.ex01 패키지를 만들고 LoginServlet3 클래스를 생성.



5. 웹 브라우저에서 서블릿으로 데이터 전송

2. login.html은 앞에서 생성한 파일을 편집해서 사용다. <form> 태그의 속성 method를 post로, action을 login3으로 수정

```
<form name="frmLogin" method="post" action="login3" encType="UTF-8">
```

method 속성을 post로 수정합니다.

매핑 이름을 login3으로 수정합니다.

3. 다음과 같이 LoginServlet3 클래스를 작성. 서블릿에서는 반드시 doPost() 메서드를 이용해서 처리.

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

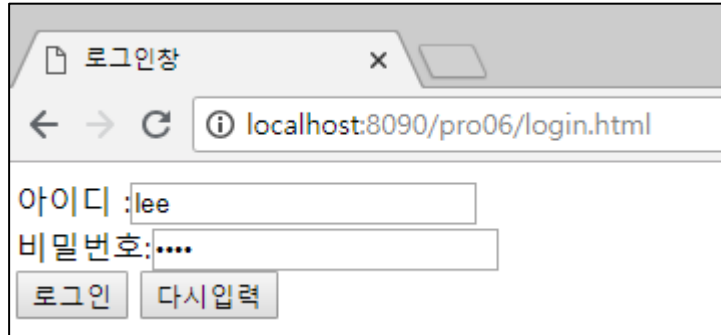
```
    request.setCharacterEncoding("utf-8");
    String user_id = request.getParameter("user_id");
    String user_pw = request.getParameter("user_pw");
    System.out.println("아이디:" + user_id);
    System.out.println("비밀번호:" + user_pw);
}
```

POST 방식으로 전송된 데이터를 처리하기 위해 doPost()를 이용합니다.

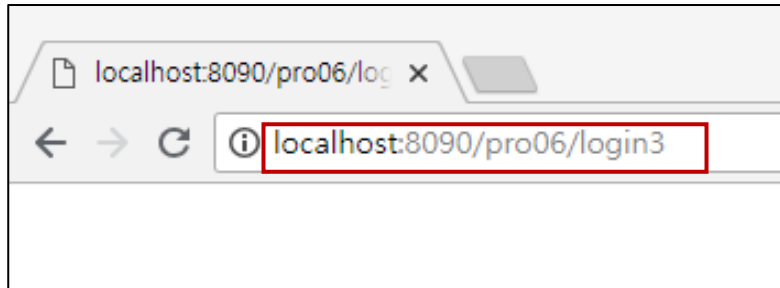
```
public void destroy() {
    System.out.println("destroy 메서드 호출");
}
}
```

5. 웹 브라우저에서 서블릿으로 데이터 전송

4. 웹 브라우저에서 로그인창을 요청한 후 ID와 비밀번호를 입력하고 로그인을 클릭.



5. 웹 브라우저에서 전송되는 데이터는 TCP/IP의 헤더에 숨겨진 채 전송되므로 브라우저의 주소창을 보면 URL 뒤에는 아무것도 표시되지 않음.

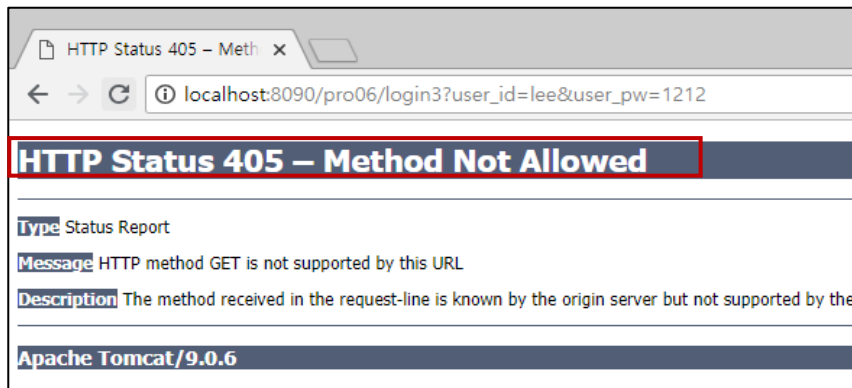


5. 웹 브라우저에서 서블릿으로 데이터 전송

- 서블릿에서는 웹 브라우저에서 전송되는 방식에 따라 doGet()이나 doPost() 메서드로 대응해서 처리해야함.

```
<form name="frmLogin" method="get" action="login3" encType="UTF-8">  
아이디 :<input type="text" name="user_id"><br>  
비밀번호:<input type="password" name="user_pw" ><br>  
<input type="submit" value="로그인"> <input type="reset" value="다시입력">  
</form>
```

```
35 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,  
36     request.setCharacterEncoding("utf-8");  
37     String user_id=request.getParameter("user_id");  
38     String user_pw=request.getParameter("user_pw");  
39     System.out.println("아이디:"+user_id);  
40     System.out.println("비밀번호:"+user_pw);  
41  
42 }
```



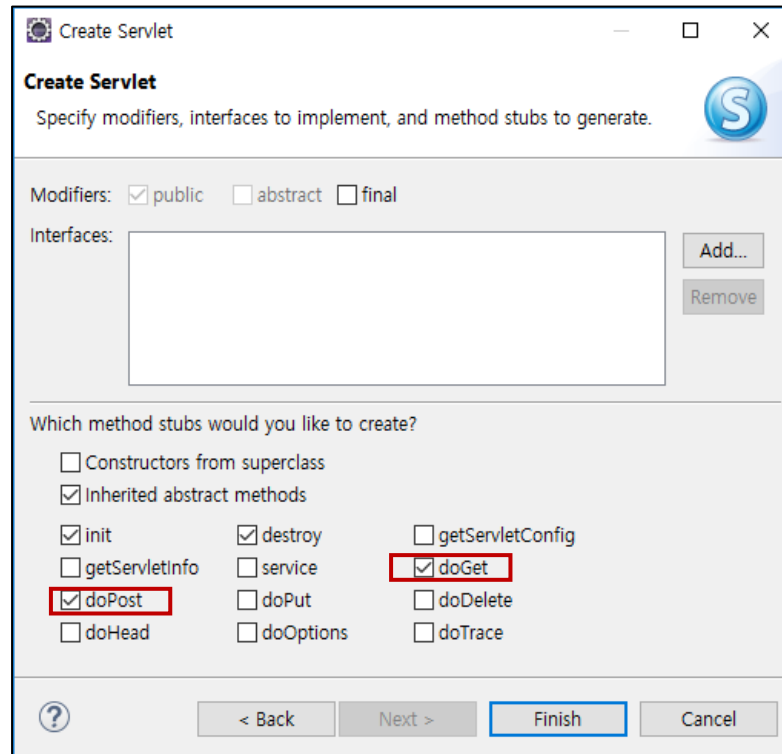
6. GET방식과 POST방식 요청 동시에 처리

1. 앞에서 실습한 login.html을 다음과 같이 수정. GET 방식으로 로그인하기 위해 method는 get으로, action은 login4로 수정.

```
<form name="frmLogin" method="get" action="login4" encType="utf-8">
```

method 속성을 get으로, action 속성을 login4로 수정합니다.

2. sec03.ex02 패키지에 LoginServlet4 서블릿을 만들 때 doGet()과 doPost()를 모두 추가



6. GET방식과 POST방식 요청 동시에 처리

3. LoginServlet4 클래스를 다음과 같이 작성합니다. doGet()과 doPost() 메서드에서 doHandle() 메서드를 재호출하여 모든 방식의 요청을 처리합니다.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    System.out.println("doGet 메서드 호출");
    doHandle(request, response);
}
```

GET 방식으로 요청 시 다시 doHandle()을 호출합니다.

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    System.out.println("doPost 메서드 호출");
    doHandle(request, response);
}
```

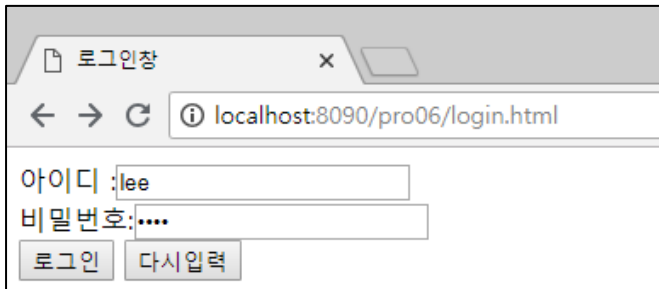
POST 방식으로 요청 시 다시 doHandle()을 호출합니다.

```
private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    request.setCharacterEncoding("utf-8");
    String user_id = request.getParameter("user_id");
    System.out.println("doHandle 메서드 호출");
    String user_pw = request.getParameter("user_pw");
    System.out.println("아이디:" + user_id);
    System.out.println("비밀번호:" + user_pw);
}
```

모든 호출 방식에 대해 처리할 수 있습니다.

6. GET방식과 POST방식 요청 동시에 처리

4. GET 방식과 POST 방식으로 각각 요청해 보겠습니다. 두 방식 모두 doHandle() 메서드로 처리한 후 결과를 출력합니다.



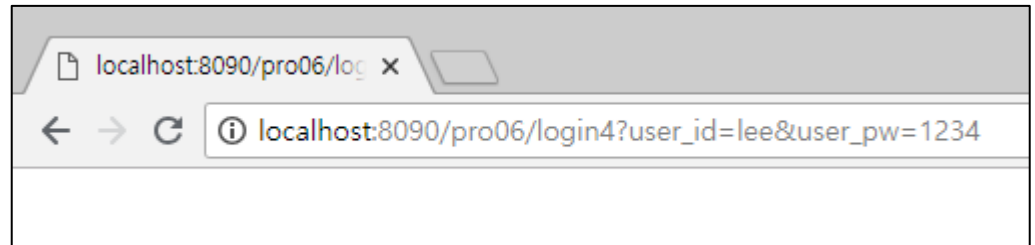
로그인창

← → ↻ ⓘ localhost:8090/pro06/login.html

아이디 :lee

비밀번호:....

로그인 다시입력



```
8월 24, 2018 1:05:45 오후 (
정보: Server startup in 24
init 메서드 호출
doGet 메서드 호출
doHandle 메서드 호출
아이디:lee
비밀번호:1234
```

6. GET방식과 POST방식 요청 동시에 처리

5. 로그인창의 method 속성을 post로 변경한 후 요청하면 doHandle() 메서드로 처리한 후 결과를 출력

코드 pro06/WebContent/login.html

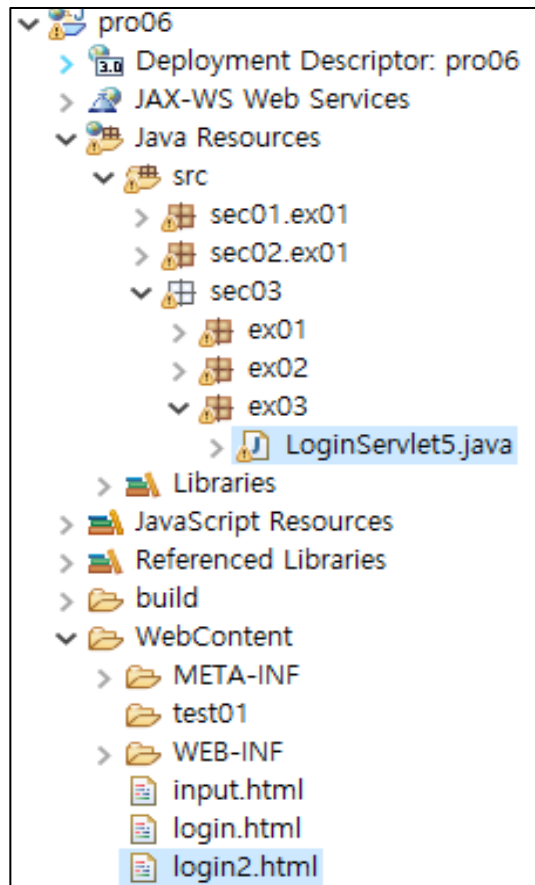
```
<form name="frmLogin" method="post" action="login4" encType="utf-8">
```

```
doPost 메서드 호출  
doHandle 메서드 호출  
아이디:lee  
비밀번호:1212
```


7. 자바스크립트로 서블릿에 요청

- 실무에선 자바스크립트에서 먼저 입력한 값에 대해서 유효성 검사를 한 후 자바스크립트에서 서블릿에 요청함.

1. 다음과 같이 sec03.ex03 패키지에 LoginServlet5 클래스를 생성하고 login2.html을 추가로 생성.



7. 자바스크립트로 서버릿에 요청

2. 다음과 같이 login2.html을 작성. 자바스크립트 함수에서 <form> 태그에 접근하여 값 입력 여부를 체크한 후 action 속성에 전송할 서버릿 이름을 지정

```
<html>
<head>
  <meta charset="UTF-8">
  <script type="text/javascript">
    function fn_validate() {
      var frmLogin = document.frmLogin;
      var user_id = frmLogin.user_id.value;
      var user_pw = frmLogin.user_pw.value;

      if ((user_id.length == 0 || user_id == "") || (user_pw.length == 0 || user_pw
      == "")) {
        alert("아이디와 비밀번호는 필수입니다.");
      } else {
        frmLogin.method = "post";
        frmLogin.action = "login5";
        frmLogin.submit();
      }
    }
  </script>
  <title>로그인창</title>
</head>

<body>
  <form name="frmLogin" method="post" action="login" encType="UTF-8">
    아이디 :<input type="text" name="user_id"><br>
    비밀번호:<input type="password" name="user_pw"><br>
    <input type="button" onClick="fn_validate()" value="로그인">
    <input type="reset" value="다시 입력">
    <input type="hidden" name="user_address" value="서울시 성북구" />
  </form>
</body>
</html>
```

Annotations:

- <form> 태그의 name 속성으로 <form> 태그 객체를 받아 옵니다.
- <form> 태그 내 <input> 태그의 name 속성으로 입력한 ID와 비밀번호를 받아 옵니다.
- <form> 태그의 전송 방식을 post로 설정합니다.
- action 속성을 서버릿 매핑 이름인 login5로 설정합니다.
- 자바스크립트에서 서버릿으로 전송합니다.
- <hidden> 태그를 이용해 화면에는 보이지 않게 하면서 값을 서버릿으로 전송합니다.

7. 자바스크립트로 서블릿에 요청

3. LoginServlet5 클래스를 다음과 같이 작성.

```
...  
@WebServlet("/login5")  
public class LoginServlet5 extends HttpServlet  
{  
    public void init()  
    {  
        System.out.println("init 메서드 호출");  
    }  
}
```

```
protected public void doPost(HttpServletRequest request, HttpServletResponse response)  
    throws ServletException, IOException  
{  
    request.setCharacterEncoding("utf-8");  
    response.setContentType("text/html;charset=utf-8");  
    PrintWriter out = response.getWriter();  
    String id = request.getParameter("user_id");  
    String pw = request.getParameter("user_pw");  
    String address = request.getParameter("user_address");  
    System.out.println("아이디   : " + id);  
    System.out.println("비밀번호 : " + pw);
```

〈hidden〉 태그로 전송된 값을 받아 옵니다.

```
String data = "<html>";  
data += "<body>";  
data += "아이디 : " + id;  
data += "<br>";  
data += "비밀번호: " + pw;  
data += "<br>";  
data += "주소 : " + address;  
data += "</html>";  
data += "</body>";  
out.print(data);
```

전송된 값을 웹 브라우저로 출력합니다.

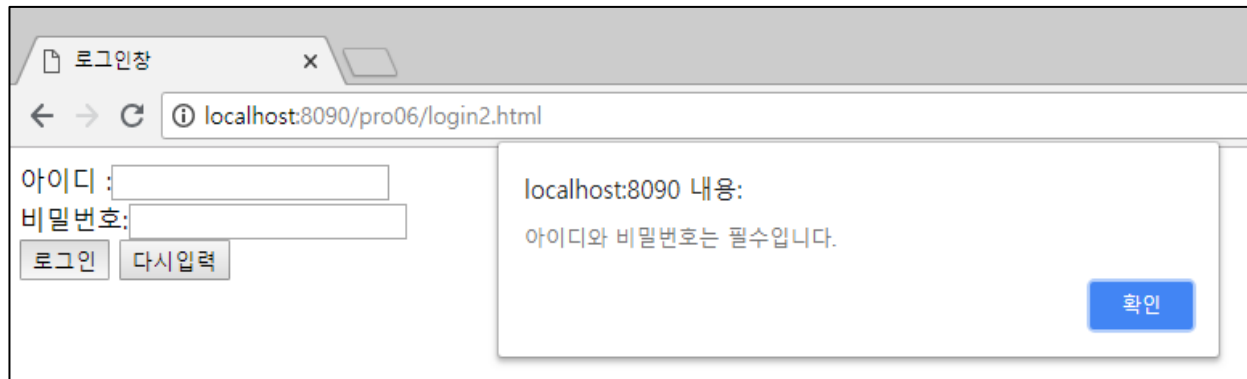
```
}
```

```
public void destroy() {  
    System.out.println("destroy 메서드 호출");  
}
```

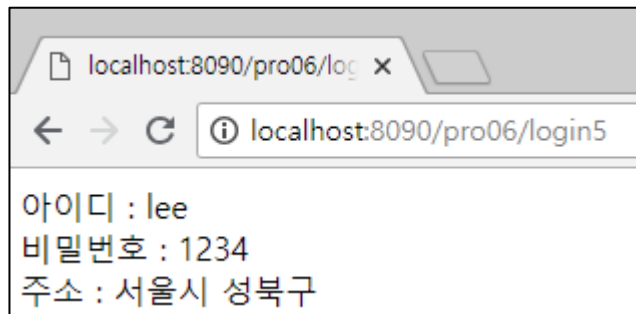
```
}
```

7. 자바스크립트로 서블릿에 요청

4. <http://localhost:8090/pro06/login2.html>로 요청. ID와 비밀번호를 입력하지 않고 로그인을 클릭하면 오류 창이 나타남.



아이디와 비밀번호를 정상적으로 입력한 경우



8. 서블릿을 이용한 여러 가지 실습 예제

• 8.1 서블릿에 로그인 요청 시 유효성 검사하기

문제: ID를 정상적으로 입력했을 때는 로그인 메시지를 표시하고, ID를 입력하지 않았을 때는 다시 로그인하라는 메시지를 표시하도록 작성하시오.

1. test01 폴더에 login.html을 만들고 다음과 같이 작성합니다.

코드 6-16 pro06/WebContent/test01/login.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>로그인 창</title>
</head>
<body>
  <form name="frmLogin" method="post" action="/pro06/loginTest" enctype="UTF-8">
    아이디 : <input type="text" name="user_id"><br>
    비밀번호:<input type="password" name="user_pw"><br>
    <input type="submit" value="로그인"> <input type="reset" value="다시 입력">
  </form>
</body>
</html>
```

test01 폴더에 위치한 HTML에서 서블릿에 요청하므로 action 속성에서 서블릿 매핑 이름 앞에 프로젝트 이름 /pro06을 붙여주어야 합니다.

텍스트 박스에 입력한 ID를 name 속성인 user_id로 전송합니다.

텍스트 박스에 입력한 비밀번호를 name 속성인 user_pw로 전송합니다.

8. 서블릿을 이용한 여러 가지 실습 예제

2. LoginTest 클래스를 다음과 같이 작성합니다. ID나 비밀번호를 제대로 입력하지 않으면 오류 메시지를 출력한 후 다시 로그인창으로 이동합니다.

코드 6-17 pro06/src/sec04/ex01/LoginTest.java

```
package sec04.ex01;
```

```
@WebServlet("/loginTest")
```

```
public class LoginTest extends HttpServlet
```

```
{
```

```
    public void init()
```

```
    {
```

```
        System.out.println("init 메서드 호출");
```

```
    }
```

```
protected public void doPost(HttpServletRequest request, HttpServletResponse response)
```

```
    throws IOException, ServletException{
```

```
        request.setCharacterEncoding("utf-8");
```

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

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

```
        String id = request.getParameter("user_id");
```

```
        String pw = request.getParameter("user_pw");
```

```
        System.out.println("아이디   : "+ id);
```

```
        System.out.println("패스워드 : "+ pw);
```

8. 서블릿을 이용한 여러 가지 실습 예제

```
if(id!= null &&(id.length()!=0)) {
    out.print("<html>");
    out.print("<body>");
    out.print( id +" 님!! 로그인 하셨습니다." );
    out.print("</html>");
    out.print("</body>");
}

else{
    out.print("<html>");
    out.print("<body>");
    out.print("아이디를 입력하세요!!!" ) ;
    out.print("<br>");
    out.print("<a href='http://localhost:8090/pro06/test01/loginTest.html'>login.html
                                                    로그인 창으로 이동 </a>");
    out.print("</html>");
    out.print("</body>");
}

}

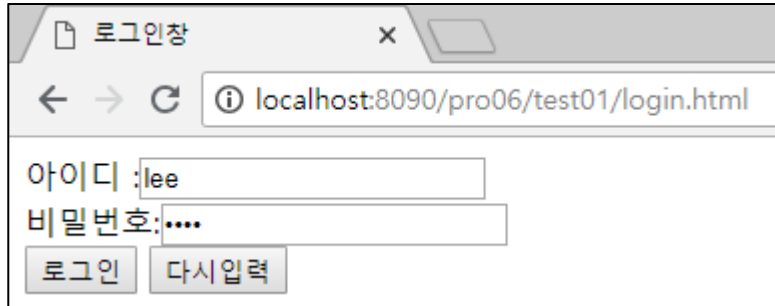
public void destroy() {
    System.out.println("destroy 메서드 호출");
}

}
```

ID와 비밀번호가 없으면 다시 로그인창으로 이동합니다.

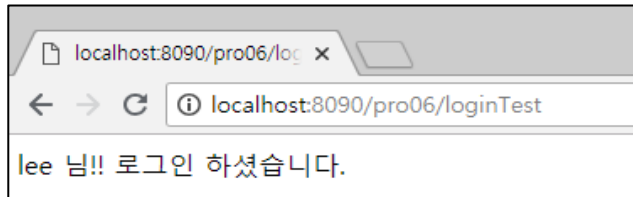
8. 서블릿을 이용한 여러 가지 실습 예제

3. `http://localhost:8090/pro06/test01/login.html`로 요청한 후 ID와 비밀번호를 정상적으로 입력하고 로그인을 클릭합니다.

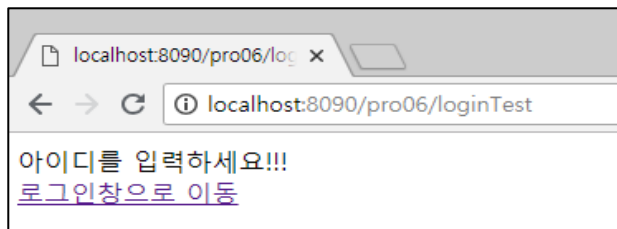


A screenshot of a web browser window titled '로그인창' (Login Window). The address bar shows 'localhost:8090/pro06/test01/login.html'. The page contains a login form with two input fields: '아이디 :lee' (ID) and '비밀번호:....' (Password). Below the fields are two buttons: '로그인' (Login) and '다시입력' (Re-enter).

4. 로그인 성공 메시지가 정상적으로 출력됩니다.



아이디를 입력하지 않은 경우



8. 서블릿을 이용한 여러 가지 실습 예제

8.2 서블릿으로 로그인 요청 시 관리자 화면 나타내기

문제: 실습 예제 1을 이용해 로그인 시 admin ID로 로그인하면 회원 관리와 회원 삭제 기능을 보여주도록 작성하시오.

1. LoginTest2 서블릿을 생성하고 다음과 같이 작성합니다.

...

```
protected public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException{
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    String id = request.getParameter("user_id");
    String pw = request.getParameter("user_pw");

    System.out.println("아이디   : "+ id);
    System.out.println("패스워드 : "+ pw);
```

8 서블릿을 이용한 여러 가지 실습 예제

```
if(id!= null &&(id.length()!=0)) {  
    if(id.equals("admin")) {  
        out.print("<html>");  
        out.print("<body>");  
        out.print( "<font size='12'>관리자로 로그인 하셨습니다!! </font>" );  
        out.print("<br>");  
        out.print("<input type=button value='회원정보 수정하기'  />");  
        out.print("<input type=button value='회원정보 삭제하기'  />");  
        out.print("</html>");  
        out.print("</body>");  
    } else {  
        out.print("<html>");  
        out.print("<body>");  
        out.print( id +" 님!! 로그인 하셨습니다." );  
        out.print("</html>");  
        out.print("</body>");  
    }  
}  
}else{  
    out.print("<html>");  
    out.print("<body>");  
    out.print("ID와 비밀번호를 입력하세요!!!" );  
    out.print("<br>");  
    out.print("<a href='http://localhost:8090/pro06/test01/login.html'>  
        로그인창으로 이동 </a>");  
    out.print("</html>");  
}
```

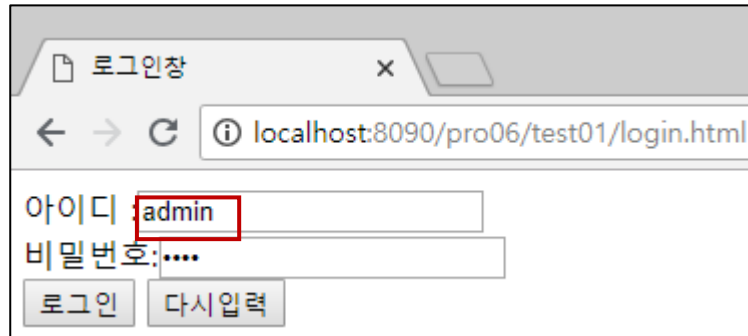
이중 `id`를 사용해 ID가 admin이면 관리자창을 보여줍니다.

관리자가 아닌 일반 사용자일 경우 로그인 성공 메시지를 보여줍니다.

8. 서블릿을 이용한 여러 가지 실습 예제

2. login.html에서 LoginTest2를 매핑하도록 수정.

3. <http://localhost:8090/pro06/login.html>로 요청한 후 ID를 admin으로 입력한 후 로그인합니다.



로그인창

← → ↻ ⓘ localhost:8090/pro06/test01/login.html

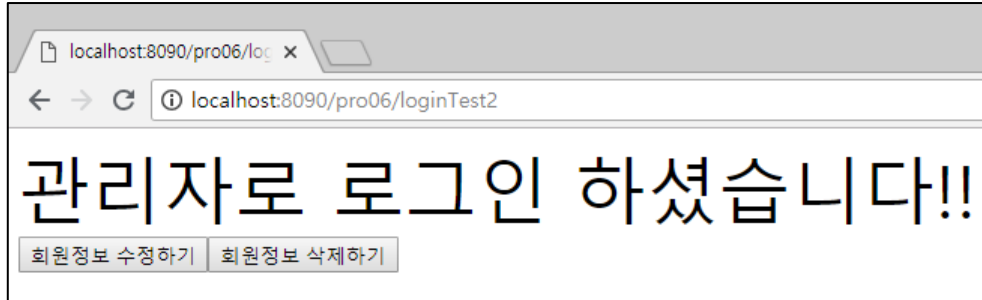
아이디: admin

비밀번호:

로그인 다시입력

8. 서블릿을 이용한 여러 가지 실습 예제

4. 다음과 같이 "관리자로 로그인 하셨습니다!!"라는 메시지가 표시됩니다.

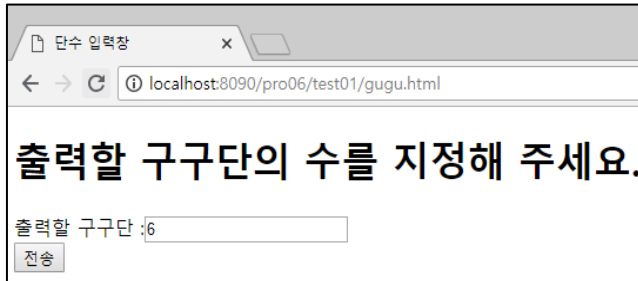


8. 서블릿을 이용한 여러 가지 실습 예제

8.3 서블릿으로 요청 시 구구단 출력하기

문제: 구구단 단수를 입력 받아 단수를 출력하시오.

1. 구구단의 단수를 입력 받는 gugu.html을 다음과 같이 작성. 단수를 입력 받아 guguTest 서블릿으로 전송
2. GuguTest 클래스를 다음과 같이 작성합니다. <table> 태그의 <tr> 태그와 자바의 for문을 이용해 구구단을 연속해서 행으로 출력합니다.
3. 출력 결과 화면에서 구구단 입력창을 요청한 후 단수를 입력합니다.
4. 전송된 단수에 대한 구구단이 브라우저에 행으로 출력됩니다.



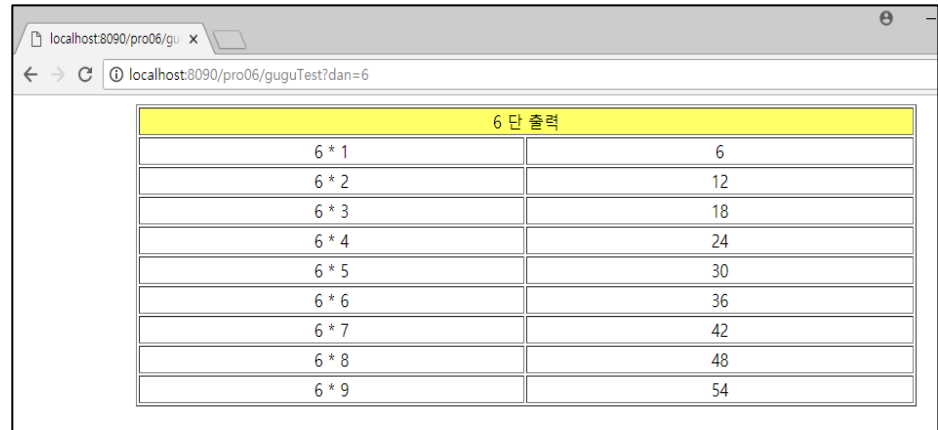
단수 입력창

localhost:8090/pro06/test01/gugu.html

출력할 구구단의 수를 지정해 주세요.

출력할 구구단 :6

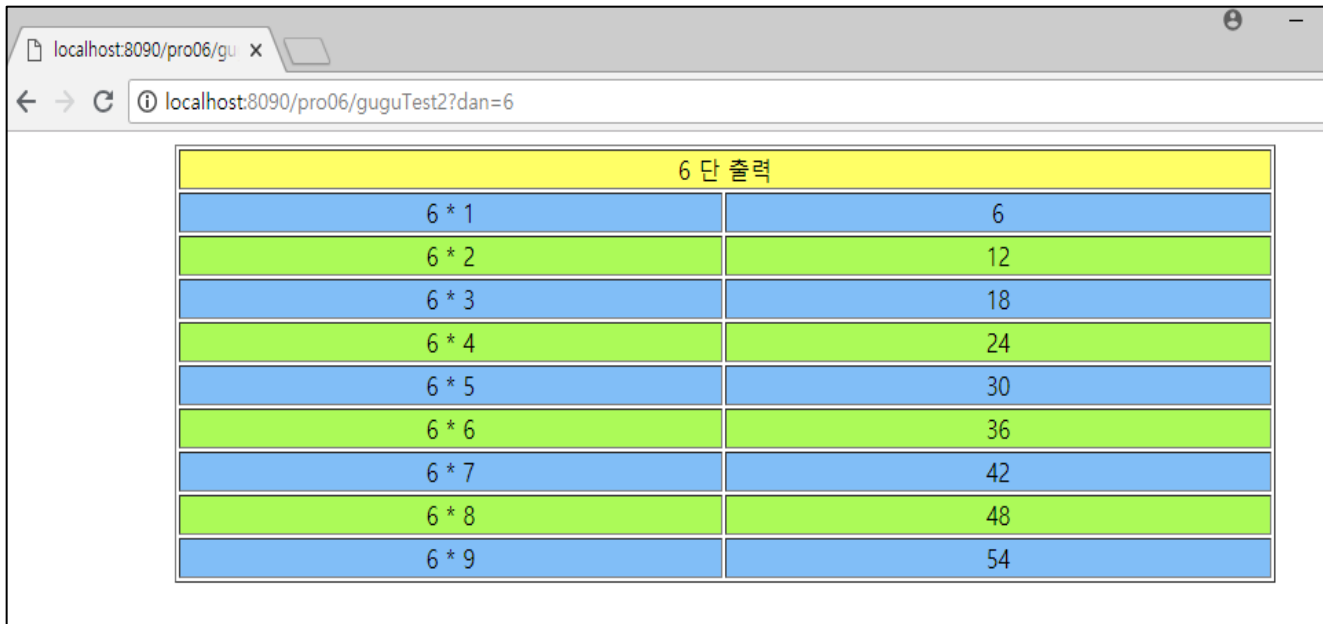
전송



6 단 출력	
6 * 1	6
6 * 2	12
6 * 3	18
6 * 4	24
6 * 5	30
6 * 6	36
6 * 7	42
6 * 8	48
6 * 9	54

8. 서블릿을 이용한 여러 가지 실습 예제

5. 이번에는 서블릿의 응답 기능을 이용해 구구단 테이블의 행 배경색을 교대로 바꾸어 보겠습니다. 다음과 같이 GuguTest2 클래스를 생성하고 코드를 작성합니다.
6. guguTest2 서블릿을 매핑하도록 gugu.html 파일을 수정한 후 브라우저에 요청합니다. 구구단 수를 입력하면 테이블의 배경색이 교대로 바뀌는 것을 확인할 수 있습니다.



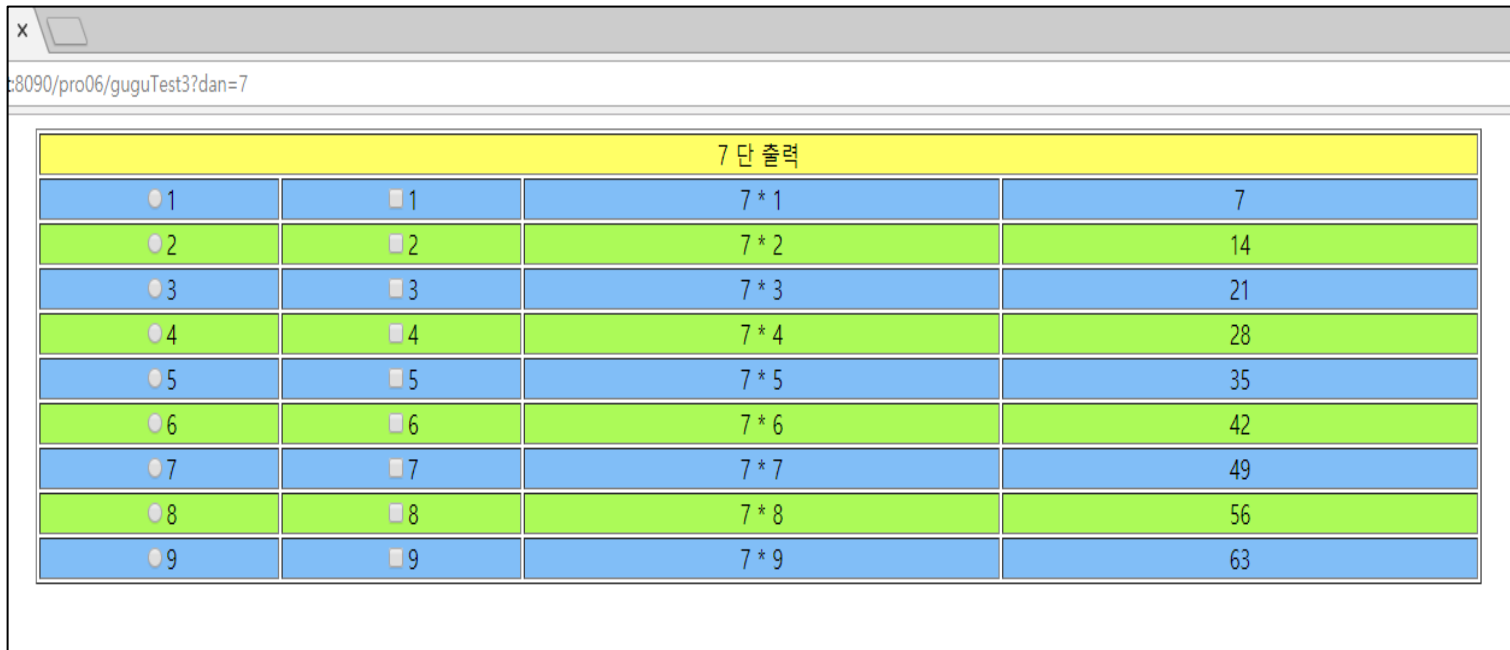
The screenshot shows a web browser window with the address bar displaying `localhost:8090/pro06/guguTest2?dan=6`. The main content area displays a table titled "6 단 출력" (6th Grade Output). The table contains 9 rows of multiplication problems for the number 6, with alternating blue and green row backgrounds.

6 단 출력	
6 * 1	6
6 * 2	12
6 * 3	18
6 * 4	24
6 * 5	30
6 * 6	36
6 * 7	42
6 * 8	48
6 * 9	54

8. 서블릿을 이용한 여러 가지 실습 예제

7. 이번에는 서블릿의 응답 기능을 조금 더 응용해서 행마다 라디오 박스와 체크박스가 표시되도록 구현해 보겠습니다. 다음과 같이 GuguTest3 클래스를 작성합니다.

8. 다음은 실행 결과입니다.



7 단 출력			
<input type="radio"/> 1	<input type="checkbox"/> 1	7 * 1	7
<input type="radio"/> 2	<input type="checkbox"/> 2	7 * 2	14
<input type="radio"/> 3	<input type="checkbox"/> 3	7 * 3	21
<input type="radio"/> 4	<input type="checkbox"/> 4	7 * 4	28
<input type="radio"/> 5	<input type="checkbox"/> 5	7 * 5	35
<input type="radio"/> 6	<input type="checkbox"/> 6	7 * 6	42
<input type="radio"/> 7	<input type="checkbox"/> 7	7 * 7	49
<input type="radio"/> 8	<input type="checkbox"/> 8	7 * 8	56
<input type="radio"/> 9	<input type="checkbox"/> 9	7 * 9	63