

---

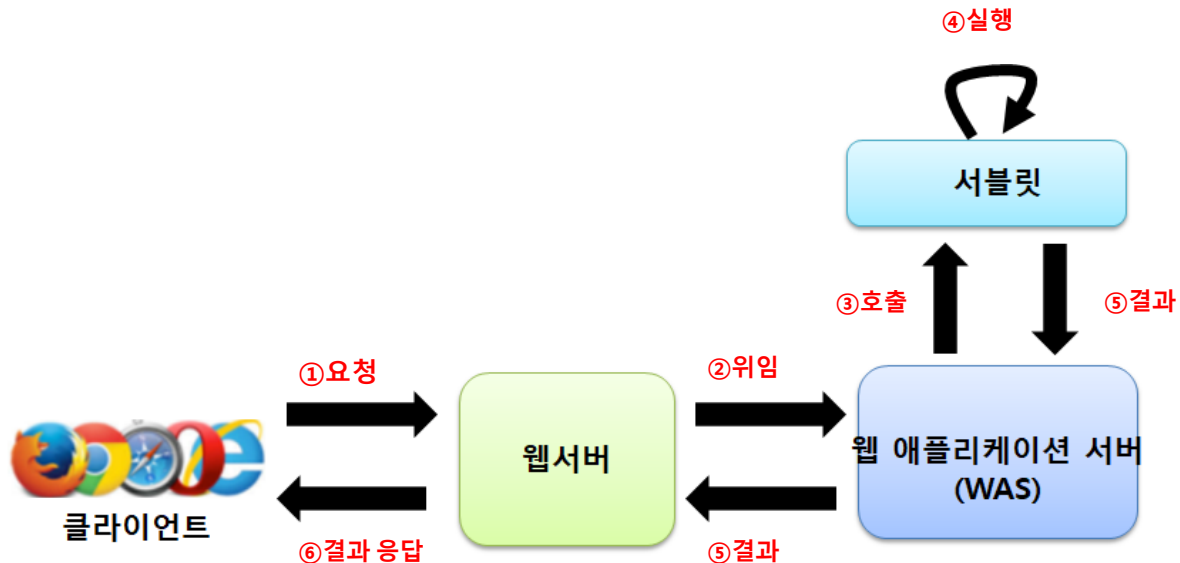
# 서블릿 이해하기

## 5 . 1 서블릿이란

- 서블릿이란?

➤ 서버 쪽에서 실행되면서 클라이언트의 요청에 따라 동적으로 서비스를 제공하는 자바 클래스

- 서블릿 동작 과정



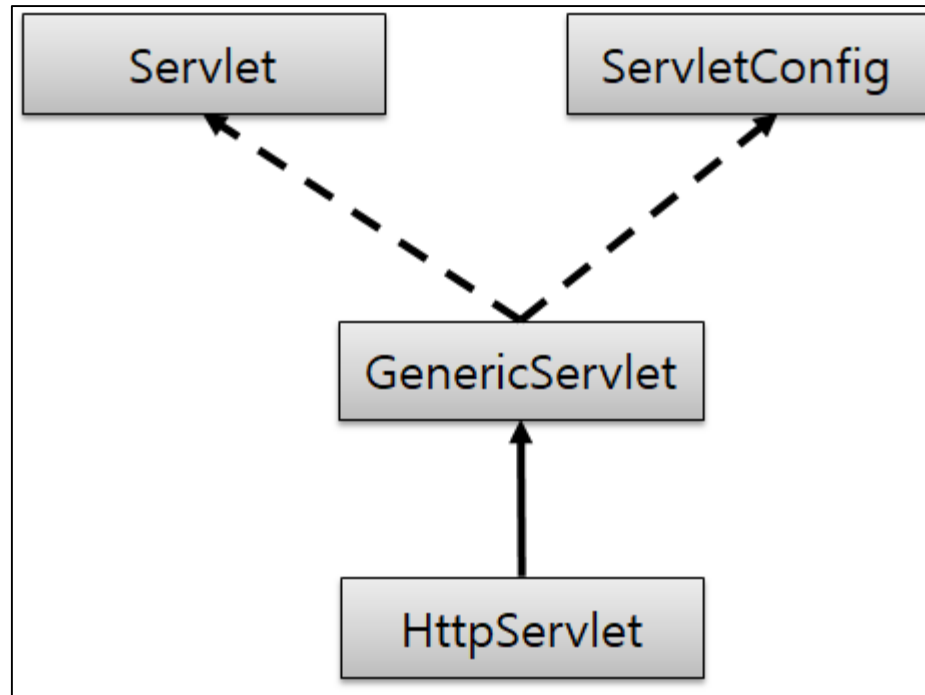
## 5 . 1 서블릿이란

- 서블릿 특징

- 서버 쪽에서 실행되면서 기능을 수행함.
- 기존의 정적인 웹 프로그램의 문제점을 보완하여 동적인 여러 가지 기능을 제공함.
- 스레드 방식으로 실행됨.
- 자바로 만들어져 자바의 특징( 객체 지향)을 가짐.
- 컨테이너에서 실행됨,
- 컨테이너 종류에 상관없이 실행됨( 플랫폼 독립적).
- 보안 기능을 적용하기 쉬움.
- 웹 브라우저에서 요청 시 기능을 수행함.

# 서블릿 API 계층 구조와 기능

- 서블릿 클래스 계층 구조



- `GenericServlet` 추상클래스는 `Servlet`과 `ServletConfig` 인터페이스를 구현함
- `HttpServlet`은 `GenericServlet` 추상클래스를 상속받음

# 서블릿 API 계층 구조와 기능

## • 서블릿 API 기능

### 서블릿 API 구성 요소 특징

서블릿 구성요소	기능
Servlet 인터페이스	<ul style="list-style-type: none"><li>• javax.servlet 패키지에 선언되어 있습니다.</li><li>• Servlet 관련 추상 메서드를 선언합니다.</li><li>• init(), service(), destroy(), getServletInfo(), getServletConfig()를 선언합니다.</li></ul>
ServletConfig 인터페이스	<ul style="list-style-type: none"><li>• javax.servlet 패키지에 선언되어 있습니다.</li><li>• Servlet 기능 관련 추상 메서드가 선언되어 있습니다.</li><li>• getInitParameter(), getInitParameterNames(), getServletContext(), getServletName()이 선언되어 있습니다.</li></ul>
GenericServlet 클래스	<ul style="list-style-type: none"><li>• javax.servlet 패키지에 선언되어 있습니다.</li><li>• 상위 두 인터페이스를 구현하여 일반적인 서블릿 기능을 구현한 클래스입니다.</li><li>• GenericServlet을 상속받아 구현한 사용자 서블릿은 사용되는 프로토콜에 따라 각 service()를 오버라이딩해서 구현합니다.</li></ul>
HttpServlet 클래스	<ul style="list-style-type: none"><li>• javax.servlet.http 패키지에 선언되어 있습니다.</li><li>• GenericServlet을 상속받아 HTTP 프로토콜을 사용하는 웹 브라우저에서 서블릿 기능을 수행합니다.</li><li>• 웹 브라우저 기반 서비스를 제공하는 서블릿을 만들 때 상속받아 사용합니다</li><li>• 요청 시 service()가 호출되면서 요청 방식에 따라 doGet()이나 doPost()가 차례 대로 호출됩니다.</li></ul>

- GenericServlet 클래스는 여러 통신 프로토콜에 대한 서블릿 기능을 구현하는함.
- GenericServlet 클래스를 상속받는 HttpServlet 클래스는 HTTP프로토콜을 사용하는 서블릿 기능을 수행함.

# 서블릿 API 계층 구조와 기능

## HttpServlet 클래스의 여러 가지 메서드 기능

메서드	기능
protected doDelete(HttpServletRequest req, HttpServletResponse resp)	서블릿이 DELETE request를 수행하기 위해 service()를 통해서 호출됩니다.
protected doGet(HttpServletRequest req, HttpServletResponse resp)	서블릿이 GET request를 수행하기 위해 service()를 통해서 호출됩니다.
protected doHead(HttpServletRequest req, HttpServletResponse resp)	서블릿이 HEAD request를 수행하기 위해 service()를 통해서 호출됩니다.
protected doPost(HttpServletRequest req, HttpServletResponse resp)	서블릿이 POST request를 수행하기 위해 service()를 통해서 호출됩니다.
protected service (HttpServletRequest req, HttpServletResponse resp)	표준 HTTP request를 public service()에서 전달받아 doXXX() 메서드를 호출합니다.
public service (HttpServletRequest req, HttpServletResponse resp)	클라이언트의 request를 protected service()에게 전달합니다.

클라이언트 요청



public service() 호출



protected service() 호출



doXXX() 호출

# 서블릿의 생명주기 메서드

- 서블릿 생명주기(Life Cycle) 메서드

➤ 서블릿 실행 단계마다 호출되어 기능을 수행하는 콜백 메서드

## 서블릿의 생명 주기 메서드 기능

생명 주기 단계	호출 메서드	기능
초기화	init()	<ul style="list-style-type: none"><li>서블릿 요청 시 맨 처음 한 번만 호출됩니다.</li><li>서블릿 생성 시 초기화 작업을 주로 수행합니다.</li></ul>
작업 수행	doGet() doPost()	<ul style="list-style-type: none"><li>서블릿 요청 시 매번 호출됩니다.</li><li>실제로 클라이언트가 요청하는 작업을 수행합니다.</li></ul>
종료	destroy()	<ul style="list-style-type: none"><li>서블릿이 기능을 수행하고 메모리에서 소멸될 때 호출됩니다.</li><li>서블릿의 마무리 작업을 주로 수행합니다.</li></ul>



init() 와 destroy() 메서드는 생략 가능하나 doXXX() 메서드는 반드시 구현해야함.

# FirstServlet을 이용한 실습

- 서블릿 생성 과정

사용자 정의 서블릿 클래스 만들기



서블릿 생명주기 메서드 구현



서블릿 매핑 작업



웹 브라우저에서 서블릿 매핑 이름으로 요청하기



# FirstServlet을 이용한 실습

- 사용자 정의 서블릿 만들기

반드시 HttpServlet 클래스를 상속받아야 합니다.

코드 사용자 정의 서블릿 형식

```
public class FirstServlet extends HttpServlet {  
    @Override  
    public void init() {  
        ...  
    }  
    @Override  
protected public void doGet(HttpServletRequest req, HttpServletResponse resp) {  
        ...  
    }  
  
    @Override  
    public void destroy() {  
        ...  
    }  
}
```



init(), doGet() 또는 doPost(), destroy() 메서드를 오버라이딩해서 구현합니다.

# FirstServlet을 이용한 실습

## FirstServlet.java 생성

```
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;
```

정보: 서버가 [16,418] 밀리초 내에  
init 메서드 호출  
doGet 메서드 호출

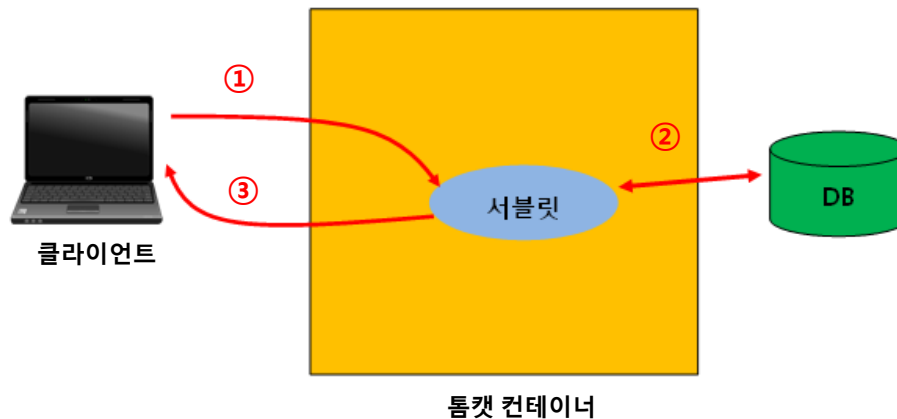
```
9
10 @WebServlet("/FirstServlet")
11 public class FirstServlet extends HttpServlet {
12     private static final long serialVersionUID = 1L;
13
14     public void init() throws ServletException {
15         System.out.println("init 메서드 호출");
16     }
17
18     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
19         throws ServletException, IOException {
20         System.out.println("doGet 메서드 호출");
21     }
22
23     protected void doPost(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         System.out.println("doPost 메서드 호출");
26     }
27     public void destroy() {
28         System.out.println("destroy 메서드 호출");
29     }
30
31 }
```

# 서블릿의 세 가지 기본 기능

- 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

# 서블릿의 세 가지 기본 기능

- 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");  
    }  
}
```

# 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 사용합니다.
String	changeSessionId()	현재 요청과 연관된 현재 세션의 id를 변경하여 새 세션 id를 반환합니다.
String	getContextPath()	요청한 컨텍스트를 가리키는 URI를 반환합니다.
Cookie[]	getCookies()	클라이언트가 현재의 요청과 함께 보낸 쿠키 객체들에 대한 배열을 반환합니다.
String	getHeader(String name)	특정 요청에 대한 헤더 정보를 문자열로 반환합니다.
Enumeration <String>	getHeaderNames	현재의 요청에 포함된 헤더의 name 속성을 enumeration으로 반환합니다.
String	getMethod()	현재 요청이 GET, POST 또는 PUT 방식 중 어떤 HTTP 요청인지 반환합니다.
String	getRequestURI()	요청한 URL의 컨텍스트 이름과 파일 경로까지 반환합니다.
String	getServletPath()	요청한 URL에서 서블릿이나 JSP 이름을 반환합니다.
HttpSession	getSession()	현재의 요청과 연관된 세션을 반환합니다. 만약 세션이 없으면 새로 만들어서 반환합니다.
String	getPathInfo()	클라이언트가 요청 시 보낸 URL과 관련된 추가 경로 정보를 반환합니다.

# 서블릿의 세 가지 기본 기능

## HttpServletResponse의 여러가지 메서드

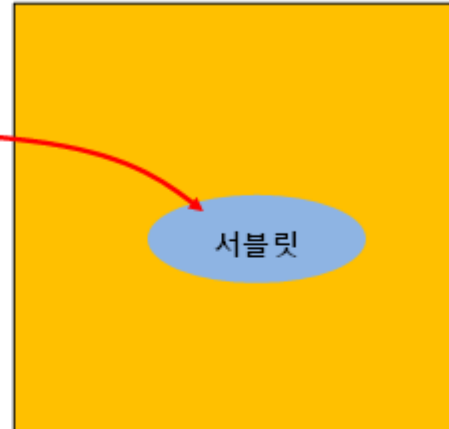
반환형	메서드 이름	기능
void	addCookie (Cookie cookie)	응답에 쿠키를 추가합니다.
void	addHeader(String name, String value)	name과 value를 헤더에 추가합니다.
String	encodeURL(String url)	클라이언트가 쿠키를 지원하지 않을 때 세션 id를 포함한 특정 URL을 인코딩합니다.
Collection <String>	getHeaderNames()	현재 응답의 헤더에 포함된 name을 얻어옵니다.
void	sendRedirect (String location)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

# < f o r m > 태그를 이용해 서블릿에 요청

- < f o r m > 태그로 서블릿에 요청하는 과정



클라이언트



톰캣 컨테이너

## 여러가지 폼태그 요소들

아이디  
 비밀번호

☒ 자바 ☐ C언어 ☐ JSP ☐ 안드로이드

☒ 짜장면 ☐ 짬뽕 ☐ 우동

글을 쓰세요.

## < f o r m > 태그를 이용해 서블릿에 요청

- < f o r m > 태그의 여러 가지 속성

아이디 : hong

비밀번호: ....

로그인    다시입력

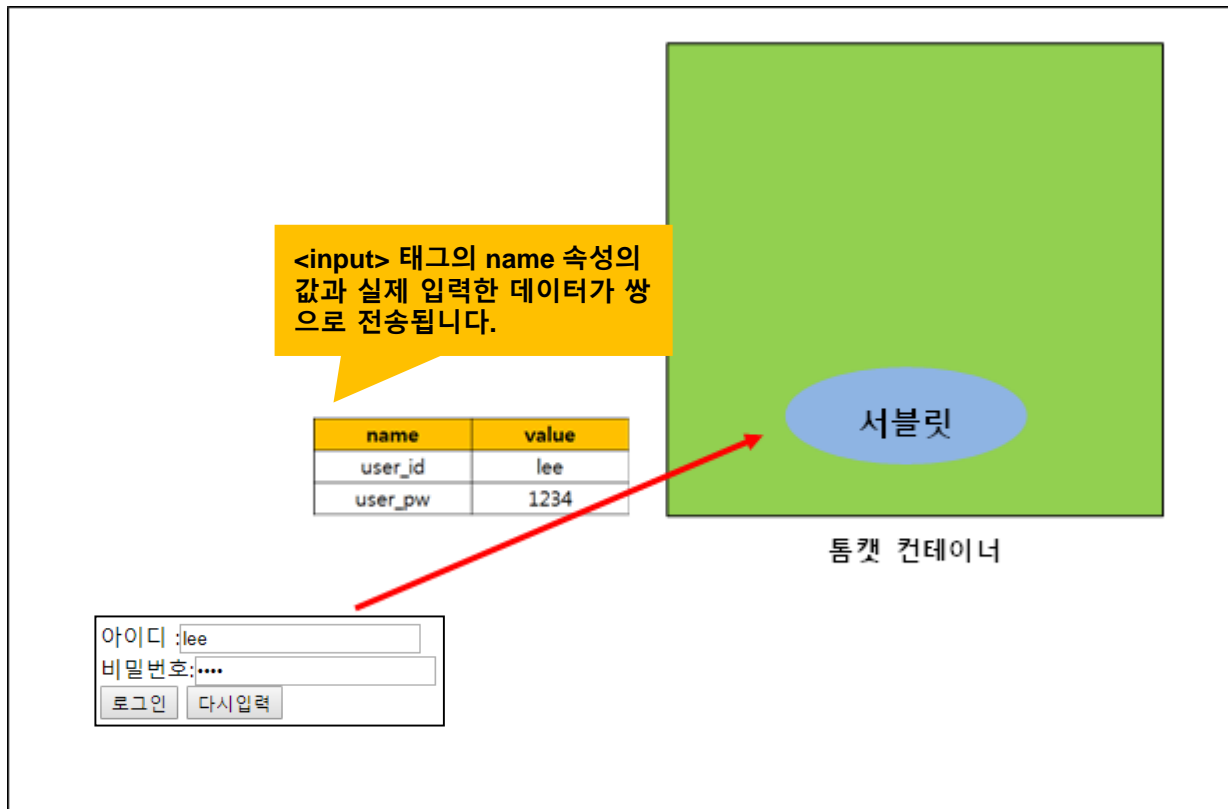


```
<form name="frmLogin" method="get" action="login" encType="UTF-8">  
  아이디 :<input type="text" name="user_id"><br>  
  비밀번호:<input type="password" name="user_pw"><br>  
  <input type="submit" value="로그인"> <input type="reset" value="다시입력">  
</form>
```



# < form > 태그를 이용해 서블릿에 요청

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



# < f o r m > 태그를 이용해 서블릿에 요청

## <form> 태그와 관련된 여러 가지 속성

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

# 서블릿에서 클라이언트의 요청 얻는 방법

- HttpServletRequest 클래스의 여러가지 메서드를 이용해서 전송된 데이터를 얻음

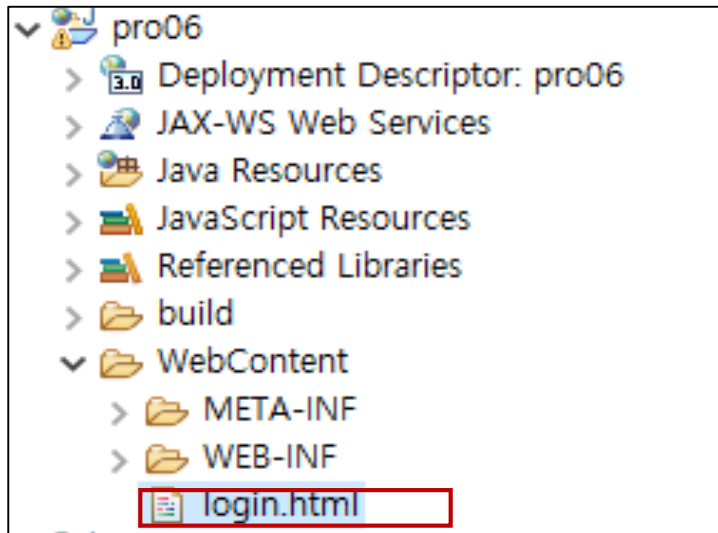
## <form> 태그로 전송된 데이터를 받아 오는 메서드

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

# 서블릿에서 클라이언트의 요청 얻는 방법

- `HttpServletRequest`로 요청 처리 실습

1. WebContent 폴더 하위에 다음과 같이 사용자 정보를 입력 받을 login.html을 생성합니다.



# 서블릿에서 클라이언트의 요청 얻는 방법

- 다음과 같이 login.html 파일을 작성합니다. 로그인창에서 ID와 비밀번호를 입력 받은 후 서블릿으로 전송하는 내용입니다.

코드 6-1 pro06/WebContent/login.html

```
<!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>
```

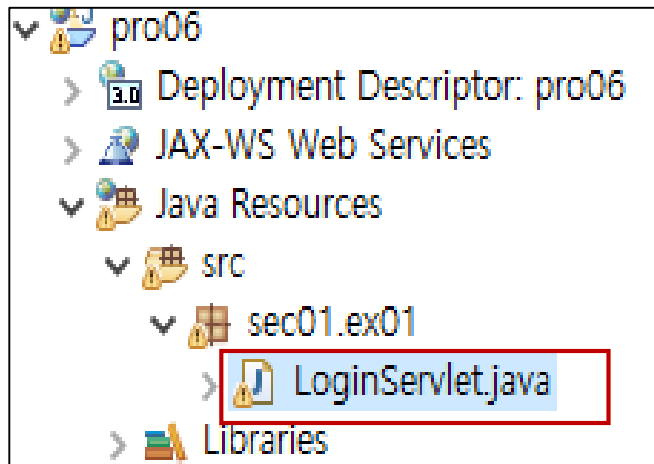
텍스트 박스에 입력된 ID를 user\_id로 전송합니다.

입력된 데이터를 서블릿 매핑 이름이 login인 서블릿으로 전송합니다.

텍스트 박스에 입력된 비밀번호를 user\_pw로 전송합니다.

# 서블릿에서 클라이언트의 요청 얻는 방법

3. 패키지를 만들고 요청을 받을 서블릿인 LoginServlet 클래스를 생성합니다 (애너테이션을 이용한 서블릿 생성 과정을 참고하세요).



# 서블릿에서 클라이언트의 요청 얻는 방법

4. 다음과 같이 LoginServlet.java 코드를 작성합니다. HttpServletRequest 클래스의 getParameter() 메서드로 전송된 ID와 비밀번호를 받아 옵니다.

코드 6-2 pro06/src/sec01/ex01/LoginServlet.java

```
package sec01.ex01;

...

@WebServlet("/login")
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");
        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);
    }

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

서블릿의 매핑 이름이 login입니다.

웹 브라우저에서 전송한 정보를 톰캣 컨테이너가 HttpServletRequest 객체를 생성한 후 doGet()으로 넘겨줍니다.

전송된 데이터를 UTF-8로 인코딩합니다.

getParameter()를 이용해 <input> 태그의 name 속성 값으로 전송된 value를 받아 옵니다.

# 서블릿에서 클라이언트의 요청 얻는 방법

5. pro06 프로젝트를 톰캣에 등록하여 실행한 후 브라우저에서 `http://localhost:8090/pro06/login.html`을 요청합니다.

아이디 :

비밀번호:

로그인

다시입력

6. 텍스트 박스에 ID와 비밀번호를 입력한 후 로그인을 클릭하면 서블릿이 ID와 비밀번호를 이클립스 콘솔에 출력합니다.

아이디 :hong

비밀번호:....

로그인

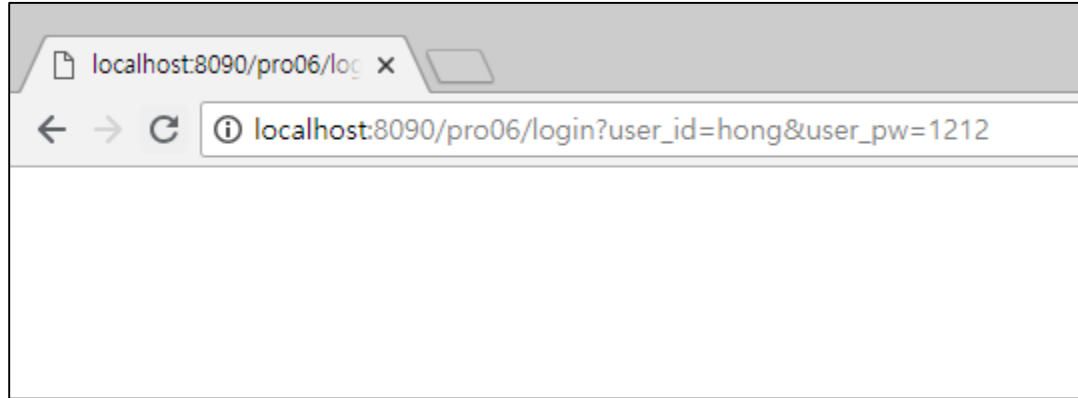
다시입력

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



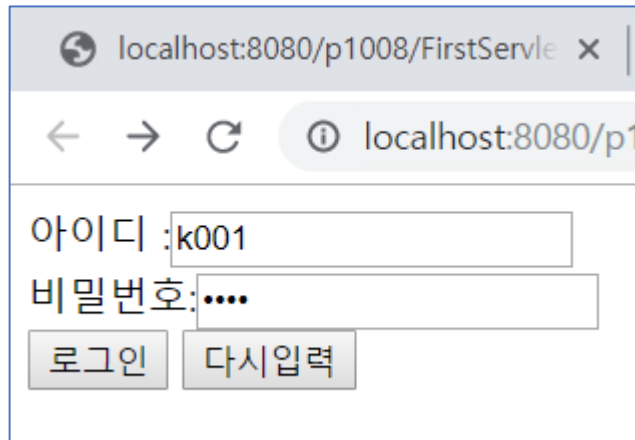
# 서블릿에서 클라이언트의 요청 얻는 방법

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



# 서블릿을 이용한 여러 가지 실습 1

다음과 같은 로그인 폼을 작성하여 서블릿으로 완성 하시오.  
일반 사용자로 로그인 한 경우



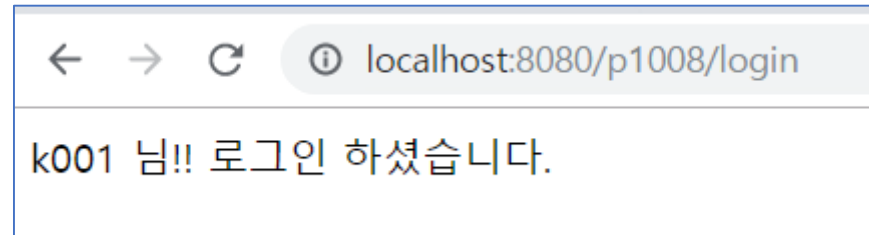
localhost:8080/p1008/FirstServle x

← → ↻ ⓘ localhost:8080/p1

아이디 : k001

비밀번호: ....

로그인 다시입력

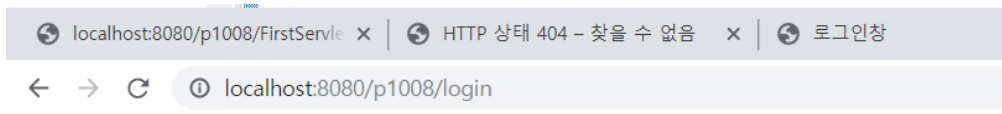
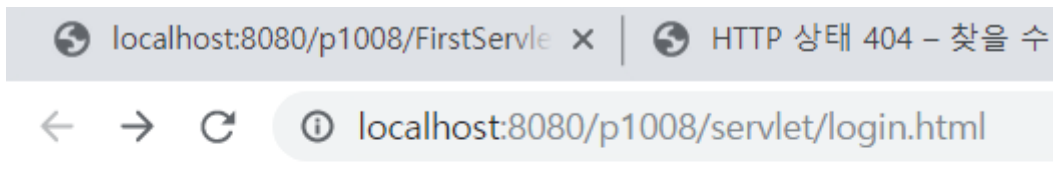


← → ↻ ⓘ localhost:8080/p1008/login

k001 님!! 로그인 하셨습니다.

# 서블릿을 이용한 여러 가지 실습 예제 1

## 관리자 'admin'으로 로그인 한 경우

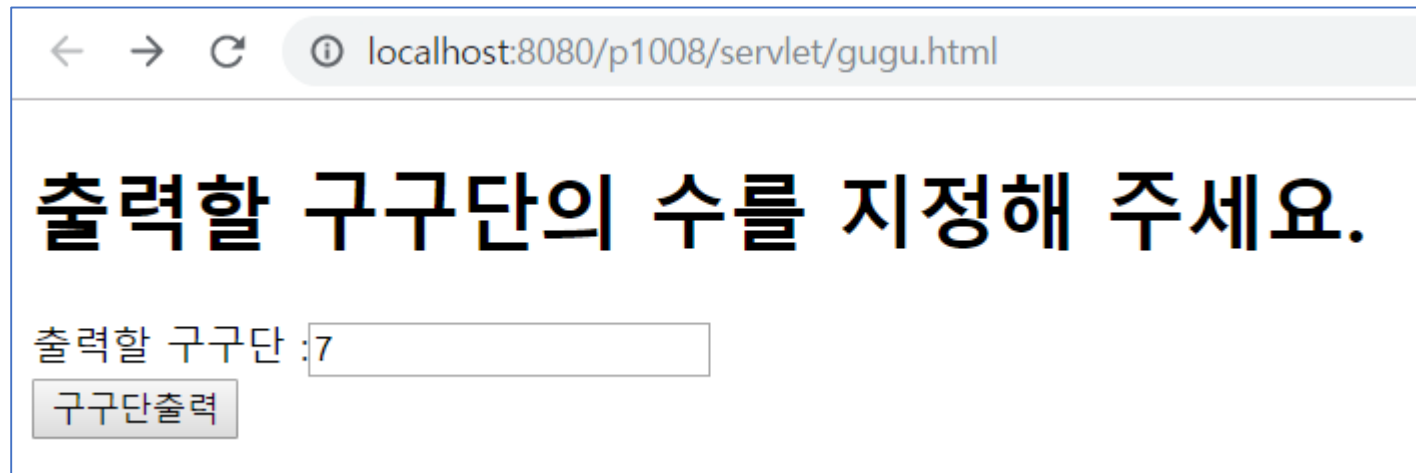


## 서블릿을 이용한 여러 가지 실습 예제 2

- 서블릿으로 요청 시 구구단 출력하기

문제: 구구단 단수를 입력 받아 단수를 출력하시오.

1. 구구단의 단수를 입력 받는 `gugu.html`을 다음과 같이 작성합니다. 단수를 입력 받아 `guguTest` 서블릿으로 전송합니다.



The screenshot shows a web browser window with the address bar displaying `localhost:8080/p1008/servlet/gugu.html`. The main content area contains the text "출력할 구구단의 수를 지정해 주세요." in large black font. Below this text is a form with the label "출력할 구구단 :" followed by a text input field containing the number "7". At the bottom left of the form is a button labeled "구구단출력".

# 서블릿을 이용한 여러 가지 실습 예제 2

2. 전송된 단수에 대한 구구단이 브라우저에 행으로 출력됩니다.

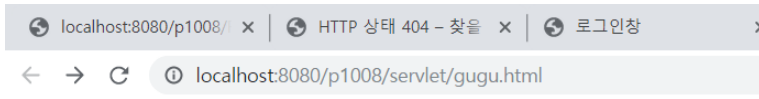


The screenshot shows a web browser window with the address bar displaying 'localhost:8090/pro06/guguTest?dan=6'. The main content area displays a table titled '6 단 출력' (6th level output). The table contains 9 rows, each representing a multiplication problem from 6 \* 1 to 6 \* 9, with the corresponding result in the second column.

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

## 서블릿을 이용한 여러 가지 실습 예제 2

2. guguTest2 서블릿을 매핑하도록 gugu.html 파일을 수정한 후 브라우저에 요청합니다. 구구단 수를 입력하면 테이블의 배경색이 교대로 바뀌는 서블릿을 작성해 보세요.



**출력할 구구단의 수를 지정해 주세요.**

출력할 구구단 :

A screenshot of a web browser window showing the output of the guguTest2 servlet. The address bar shows 'localhost:8090/pro06/guguTest2?dan=6'. The page displays a table with 9 rows of multiplication results for the number 6. The table has a yellow header row and alternating blue and green background colors for the data rows.

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