Fpt University

The Servlet Model

HTTP Methods

Form Parameters

Requests

Responses

Servlet Life Cycle

#Servlet #Video_Servlet

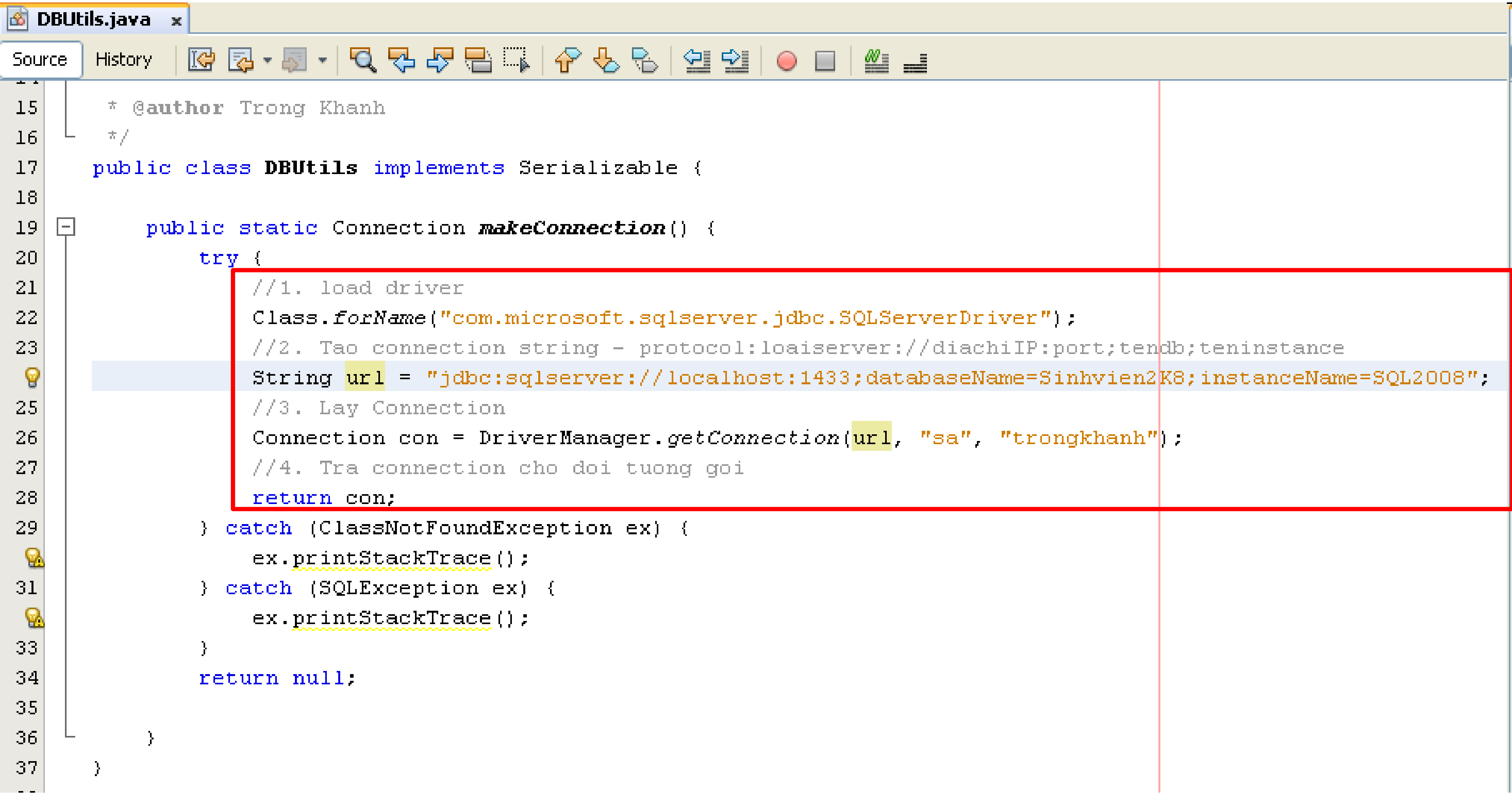
#JavaEE #MVC #MVC2

Review

- **How to connect DB using JDBC API**
  - Required
    - RDBMS: SQL Server
    - Driver Connection: sqljdbc4.jar
  - Steps
    - Load Driver
      - using **Class.forName method**
      - Driver string: **com.microsoft.sqlserver.jdbc.SQLServerDriver**
      - Exception: **ClassNotFoundException**
    - Create connection String
      - **protocol:server://ip:port;databaseName=DB[;instanceName=Instance]**
    - Open connection
      - **Connection con = DriverManager.getConnection(url, “user", “pass");**
      - Exception: **SQLException**

Review

- **How to connect DB using JDBC API**
  - Implementation

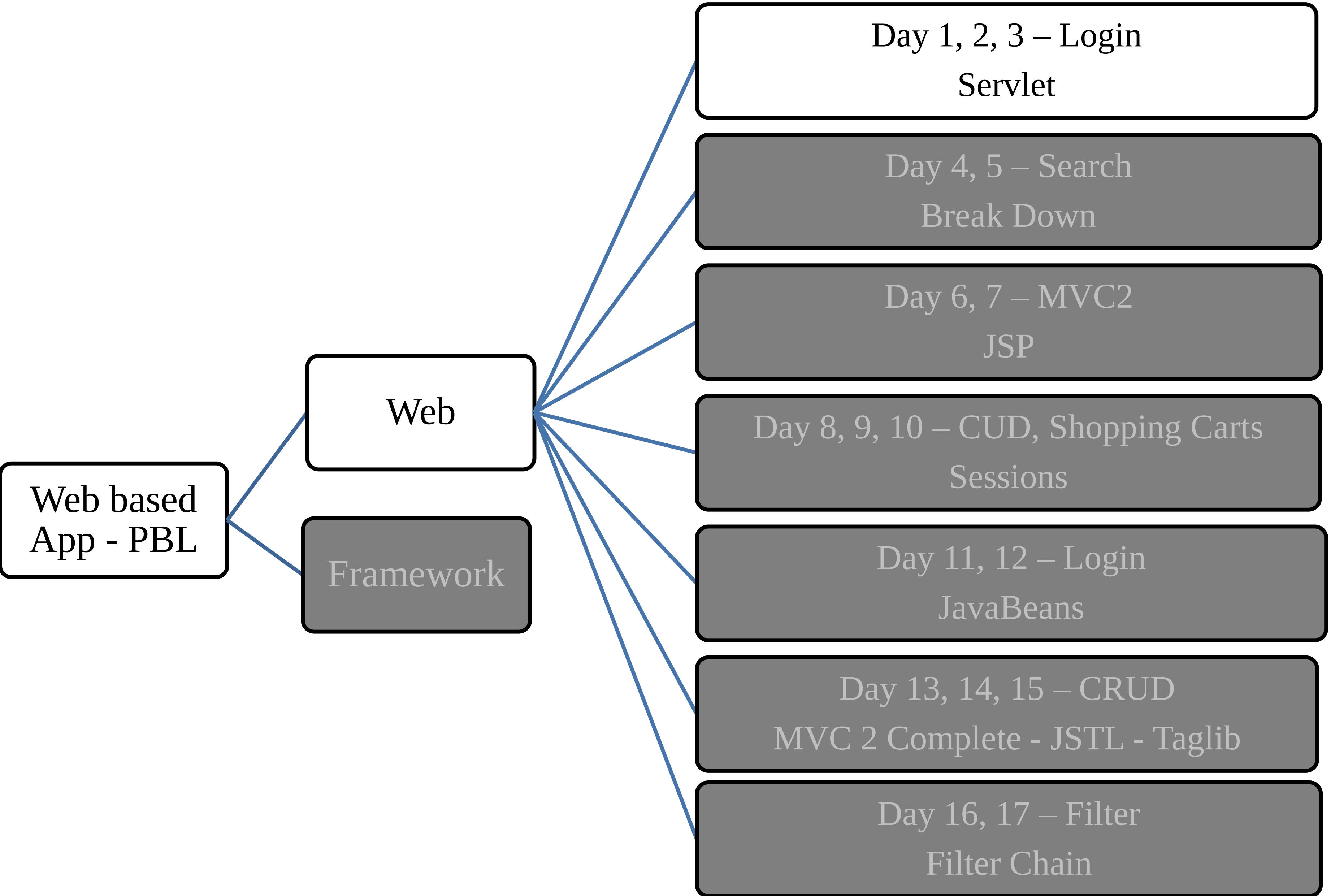


```
 * @author Trong Khanh
 */
public class DBUtils implements Serializable {

    public static Connection makeConnection() {
        try {
            //1. load driver
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            //2. Tao connection string - protocol:loaiserver://diachiIP:port;tendb;teninstance
            String url = "jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K8;instanceName=SQL2008";
            //3. Lay Connection
            Connection con = DriverManager.getConnection(url, "sa", "trongkhanh");
            //4. Tra connection cho doi tuong goi
            return con;
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        return null;

    }
}
```

Objectives

- **How to build the simple web site combining html and servlet?**
  - Http Protocol and Methods
  - What is Servlet?
  - Parameters vs. Variables
  - Servlet Life Cycle
  - Break down structure component in building web application

Objectives



Web based App - PBL

Web

Framework

Day 1, 2, 3 – Login
Servlet

Day 4, 5 – Search
Break Down

Day 6, 7 – MVC2
JSP

Day 8, 9, 10 – CUD, Shopping Carts
Sessions

Day 11, 12 – Login
JavaBeans

Day 13, 14, 15 – CRUD
MVC 2 Complete - JSTL - Taglib

Day 16, 17 – Filter
Filter Chain

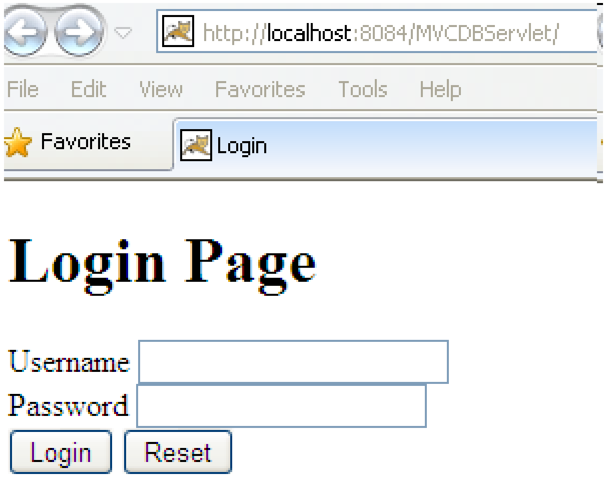
Build The Simple Web

Requirements

- Building the web application can do some following functions as
  - The user **must be authenticated** before they want to use this web site **using the DB**
  - If the user is invalid, the **message “Invalid username and password” is presented**, then the **link “Click here to try again” is shown** that **redirect the user to the login page**
  - Otherwise, **the search page** is redirected.
  - The GUI of web application is present as following

Build The Simple Web

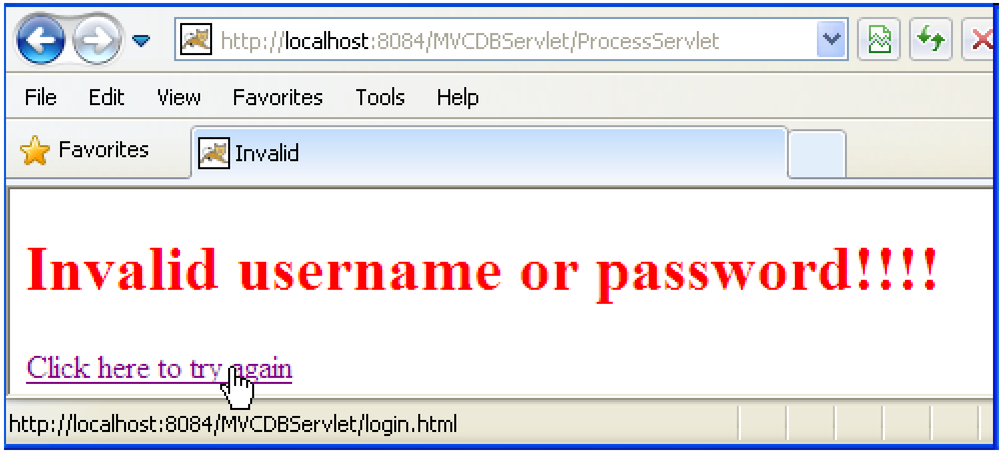
Expectation


http://localhost:8084/MVCDBServlet/
File Edit View Favorites Tools Help
Favorites
Login

Login Page

Username

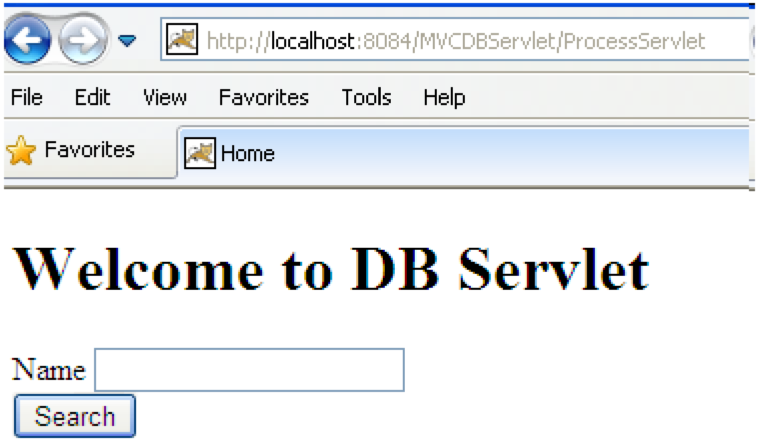
Password


http://localhost:8084/MVCDBServlet/ProcessServlet
File Edit View Favorites Tools Help
Favorites
Invalid

Invalid username or password!!!!

Click here to try again

http://localhost:8084/MVCDBServlet/login.html

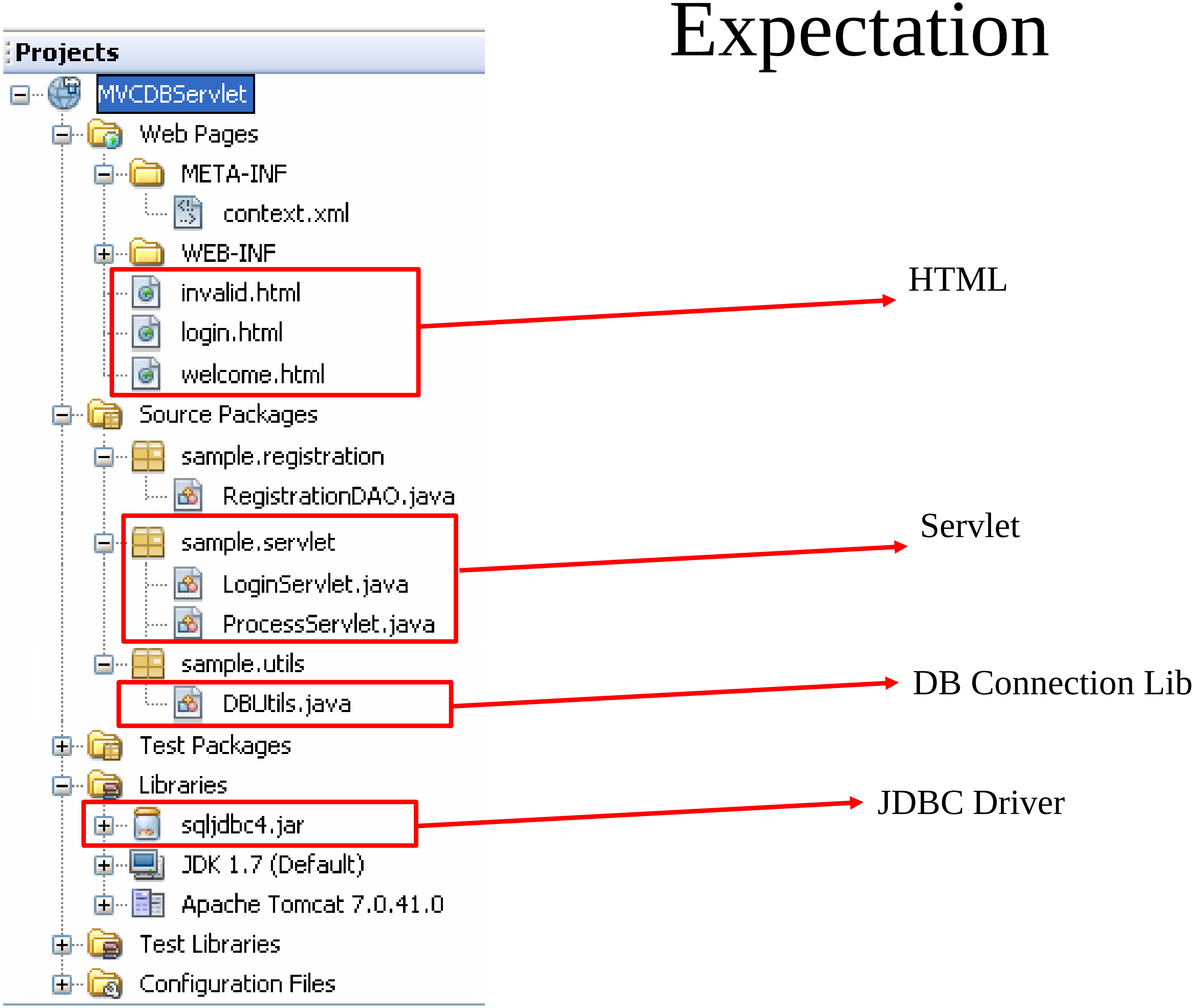

http://localhost:8084/MVCDBServlet/ProcessServlet
File Edit View Favorites Tools Help
Favorites
Home

Welcome to DB Servlet

Name

Build The Simple Web

Expectation



Projects

- MVCDBServlet
  - Web Pages
    - META-INF
      - context.xml
    - WEB-INF
    - invalid.html
    - login.html
    - welcome.html
  - Source Packages
    - sample.registration
      - RegistrationDAO.java
    - sample.servlet
      - LoginServlet.java
      - ProcessServlet.java
    - sample.utils
      - DBUtils.java
  - Test Packages
  - Libraries
    - sqljdbc4.jar
    - JDK 1.7 (Default)
    - Apache Tomcat 7.0.41.0
  - Test Libraries
  - Configuration Files

HTML

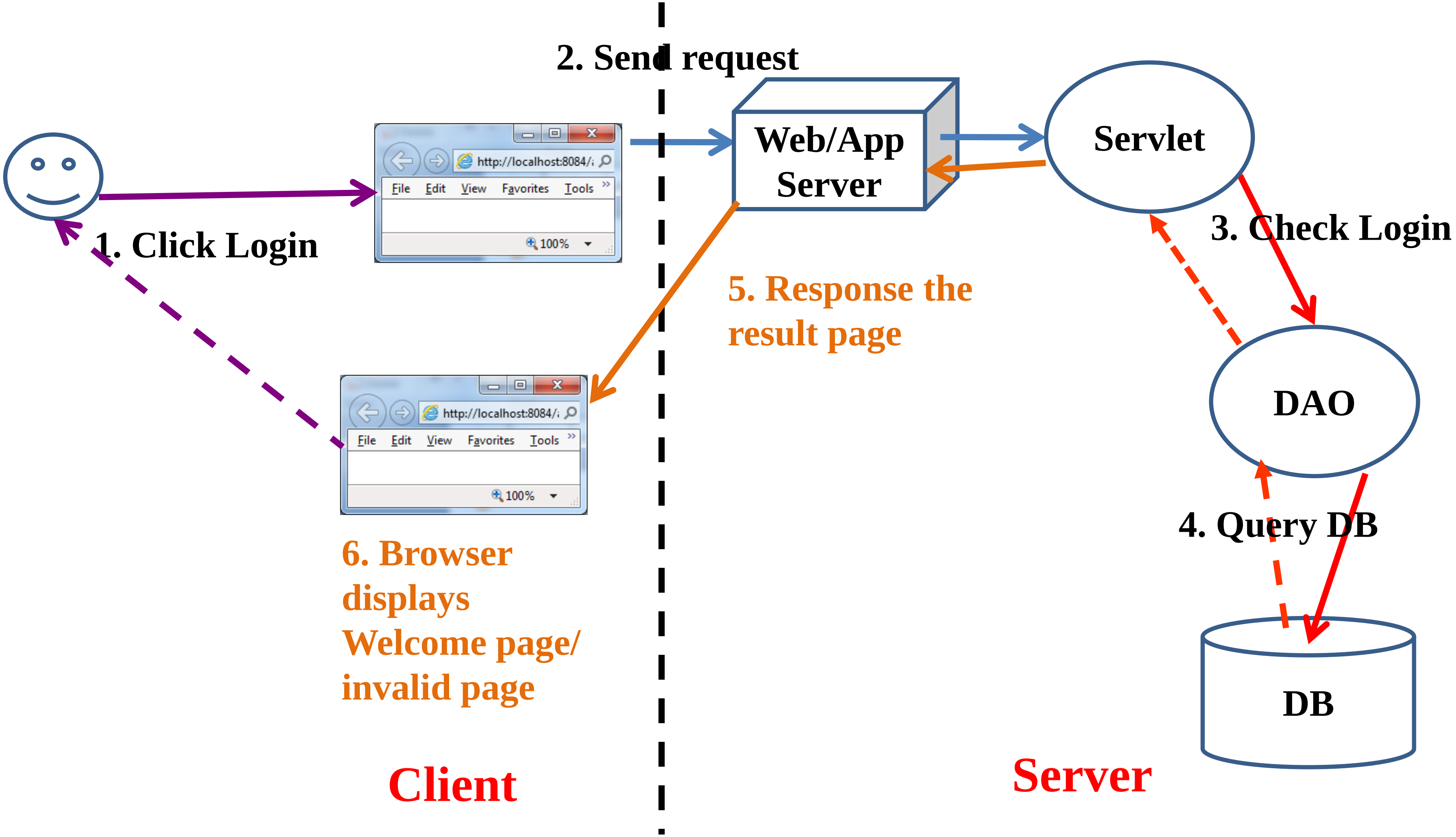
Servlet

DB Connection Lib

JDBC Driver

Build The Simple Web

Interactive Server Model



1. Click Login

2. Send request

Web/App Server

Servlet

3. Check Login

DAO

4. Query DB

DB

5. Response the result page

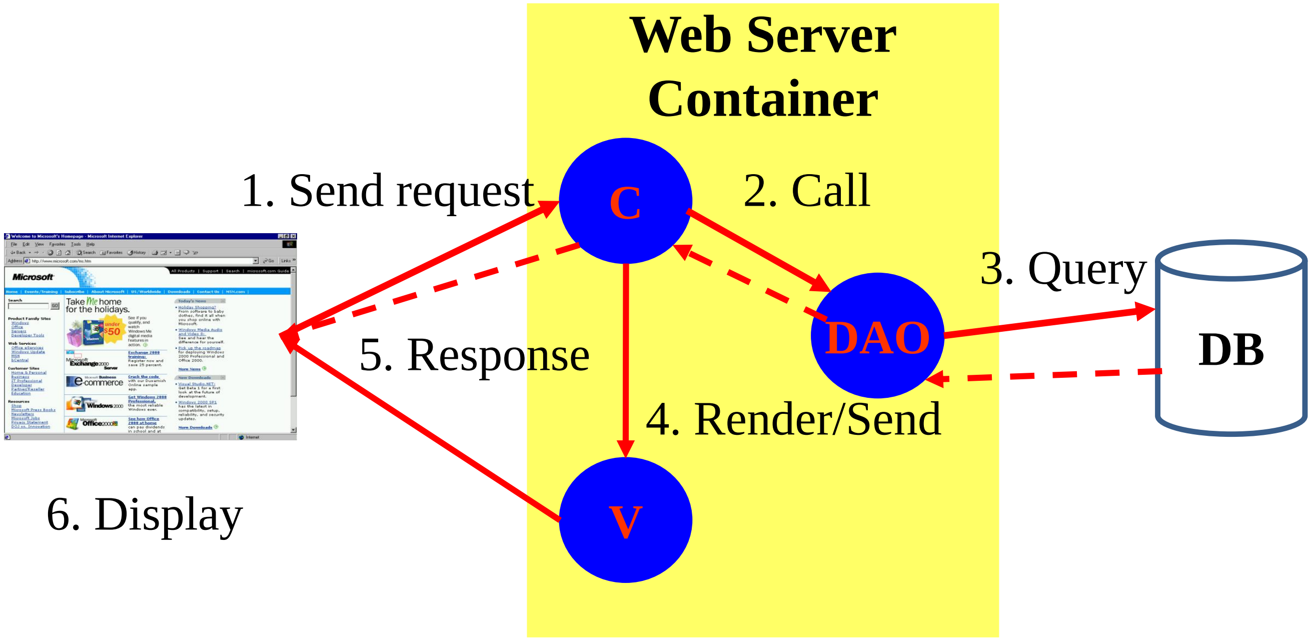
6. Browser displays Welcome page/ invalid page

Client

Server

Build The Simple Web

Abstraction



Build The Simple Web

How to Create Web Application Project

- **Requirement tools: NetBeans IDE 7.4/8.0.2/8.1**
- Create a new Web application project
  - ***Using Tomcat Server***
  - ***JavaEE 5***
  - ***Uncheck Deploy on Save***

HTML Introduction

What is HTML?

- **HTML** is a **presentation language** for **describing web pages**.
  - HTML stands for **H**yper **T**ext **M**arkup **L**anguage
  - HTML is **not** a **programming language**, it is a **markup language**
  - A markup language is a set of **markup tags**
  - HTML **uses markup tags** to describe web pages
- **HTML Documents = Web Pages**
  - HTML documents **describe web pages**
  - HTML documents **contain HTML tags** and **plain text**
  - HTML documents are also **called web pages**

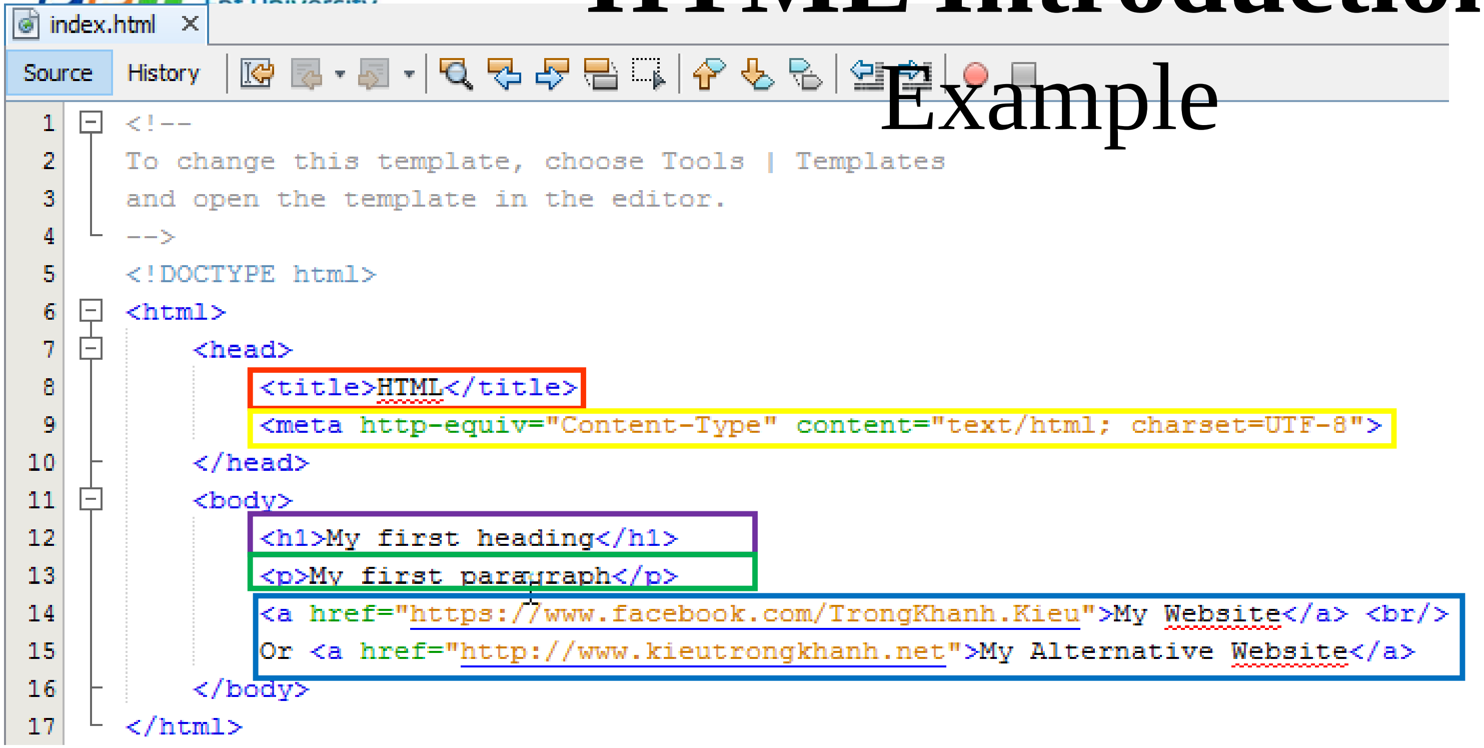
HTML Introduction

HTML Tags

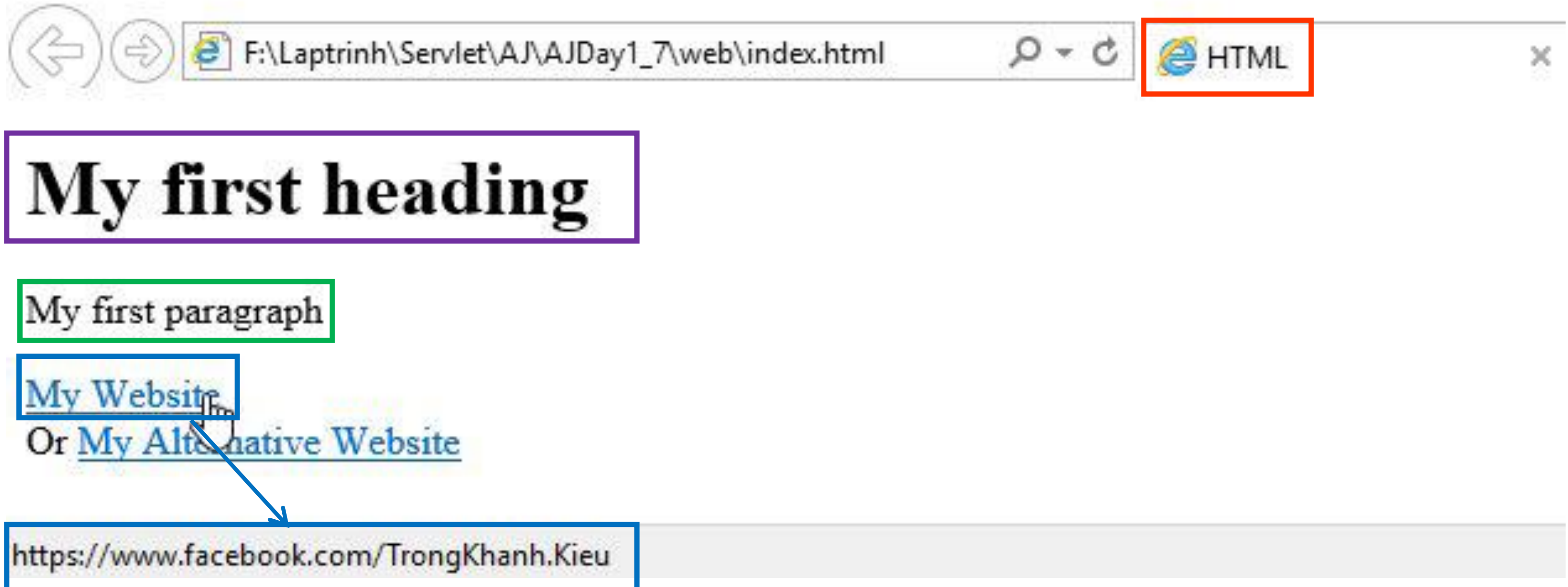
- HTML markup tags are usually called **HTML tags**
  - HTML tags are keywords surrounded by **angle brackets, that begin "<"** and **finish with ">"**, like <html>
  - HTML tags normally **come in pairs** like <b> and </b>
    - The first tag in a pair is the **start tag,** the second tag is the **end tag**
    - Start and end tags are also called **opening tags** and **closing tags**.
- **Web Browser**
  - The **purpose** of a web browser (like Internet Explorer, or Firefox, etc) is to **read HTML documents and display** them as web pages.
  - The browser **does not display** the HTML tags, but uses the tags to **interpret** the content of the page

HTML Introduction

Example



```
<!--
To change this template, choose Tools | Templates
and open the template in the editor.
-->
<!DOCTYPE html>
<html>
    <head>
        <title>HTML</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h1>My first heading</h1>
        <p>My first paragraph</p>
        <a href="https://www.facebook.com/TrongKhanh.Kieu">My Website</a> <br/>
        Or <a href="http://www.kieutrongkhanh.net">My Alternative Website</a>
    </body>
</html>
```



F:\Laptrinh\Servlet\AJ\AJDay1_7\web\index.html

My first heading

My first paragraph

My Website
Or My Alternative Website

https://www.facebook.com/TrongKhanh.Kieu

Form Parameters

HTML Form

- A form is defined on a web page **starting** with the opening tag **<form>** and **ending** with closing tag **</form>**
- **Syntax**: <form action="target" [method="HTTP method"]>
  - **action** attribute **presents value** that **contains** some **target resource** in the web application (e.g. Servlet or JSP)
  - **method** attribute **denotes** the **HTTP method** to **execute**. The **default** is to execute **HTTP GET** when the **form is submitted**
  - **Notes**: the **action** parameter **obeys** the **rules**
    - **action="targetServlet":** the browser will **assume** that targetServlet resides in the **same place the default page** as index.jsp or index.html
    - **action="/targetServlet":** the browser will **asume** the the **path at** the **root location** for specified host (http://host:port).
      - **Ex**: http://localhost:8086/targetServlet
    - **action="target?queryString"**:the request **send the data in queryString** to the URL

Form Parameters

Input Tag

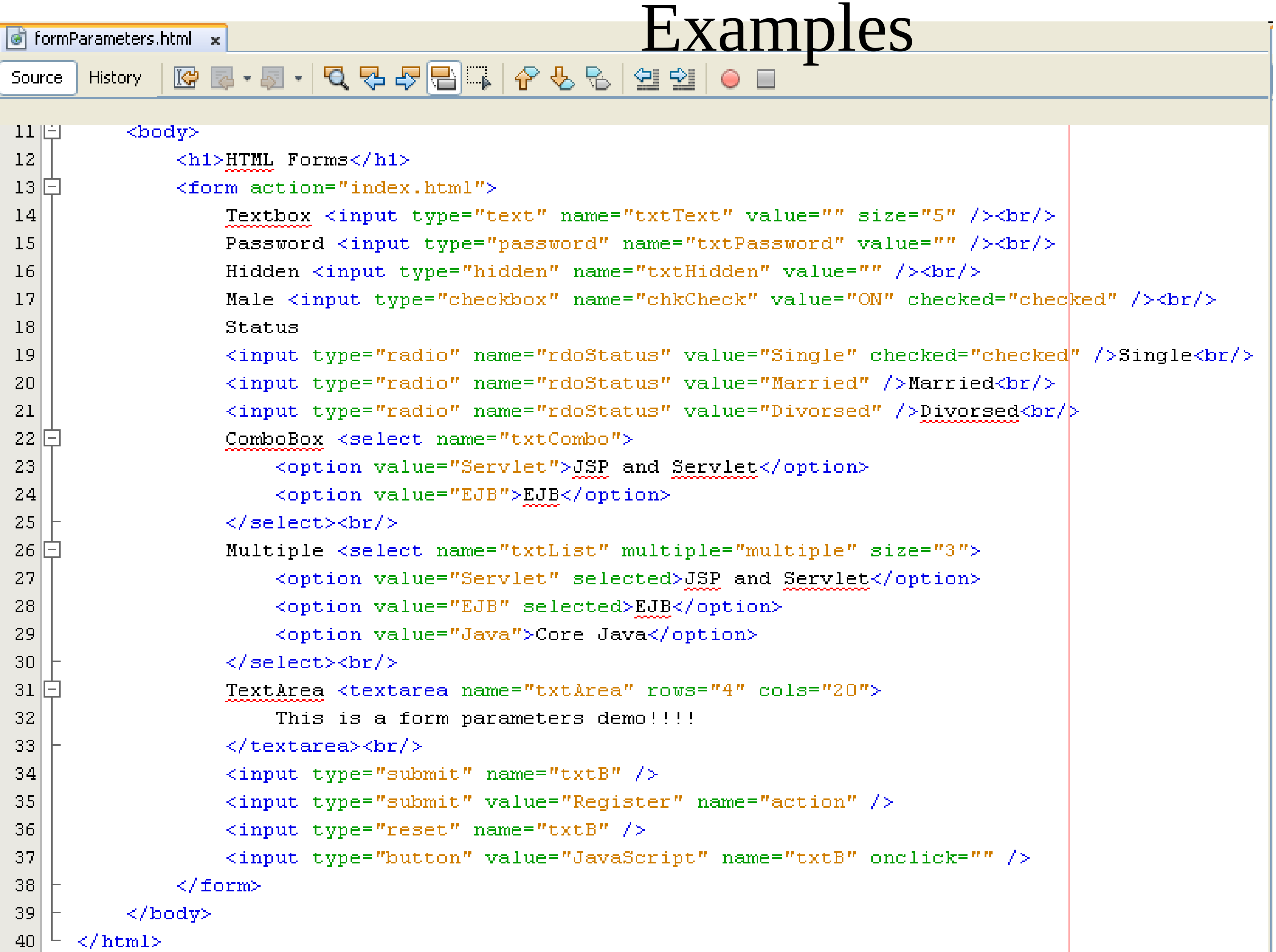
- Is used to input data
- **Syntax**: **<input type="…" [value="…" name="…"] />**
  - **type** attribute
    - Dedicates to holding a single line of text (**text**).
      - The **size** attribute specifies the width of text field in characters
      - The **maxlength** attribute controls the maximum number of characters that a user can type into the text field
    - A browser should mask the character typed in by the user (**password**)
    - Being a hidden field – is invisible (**hidden**)
    - Put one or more small boxes that can be clicked to tick or check the corresponding value denote (**checkbox**)
      - **checked="checked"** sets up the checkbox as already selected
    - The choice made is mutual exclusive (**radio**)
      - The **name attribute is crucial** to tying together a group of radio buttons
    - **Send the form data** to the URL designated by the action attribute (**submit**)
    - A request to the client browser to **reset all the values** within the form (**reset**)
    - Defining the "**custom button**" which is **connected** to **some soft of script** (**button**)
  - **name** attribute supplies the **parameter name**
  - **value** attribute supplies the **parameter value**

Form Parameters
Select & Text Area Tag

- HTML Forms – select tag
  - Sets up a **list of values to choose** (combo box or pop-up menu, or list box)
  - **Syntax**: **<select name="…" [size="…" multiple] >**
    <option value="…" [selected]>…</option>
    …
    </select>
    - **option** tag
      - The user-visible text goes between opening and closing option tag
      - The value attribute passes the value in the parameter
    - multiple attribute presents the control that can choose more than one
- HTML Forms – textarea tag
  - Presents **multiple line of text**
  - **Syntax: <textarea name="…" rows="…" cols="…">**
    …
    </textarea>
    - The text value put in opening and closing tag is passed as the parameter value to server
    - **rows** present the number of visible lines
    - **cols** present the number of characters to displayed across the width of the area

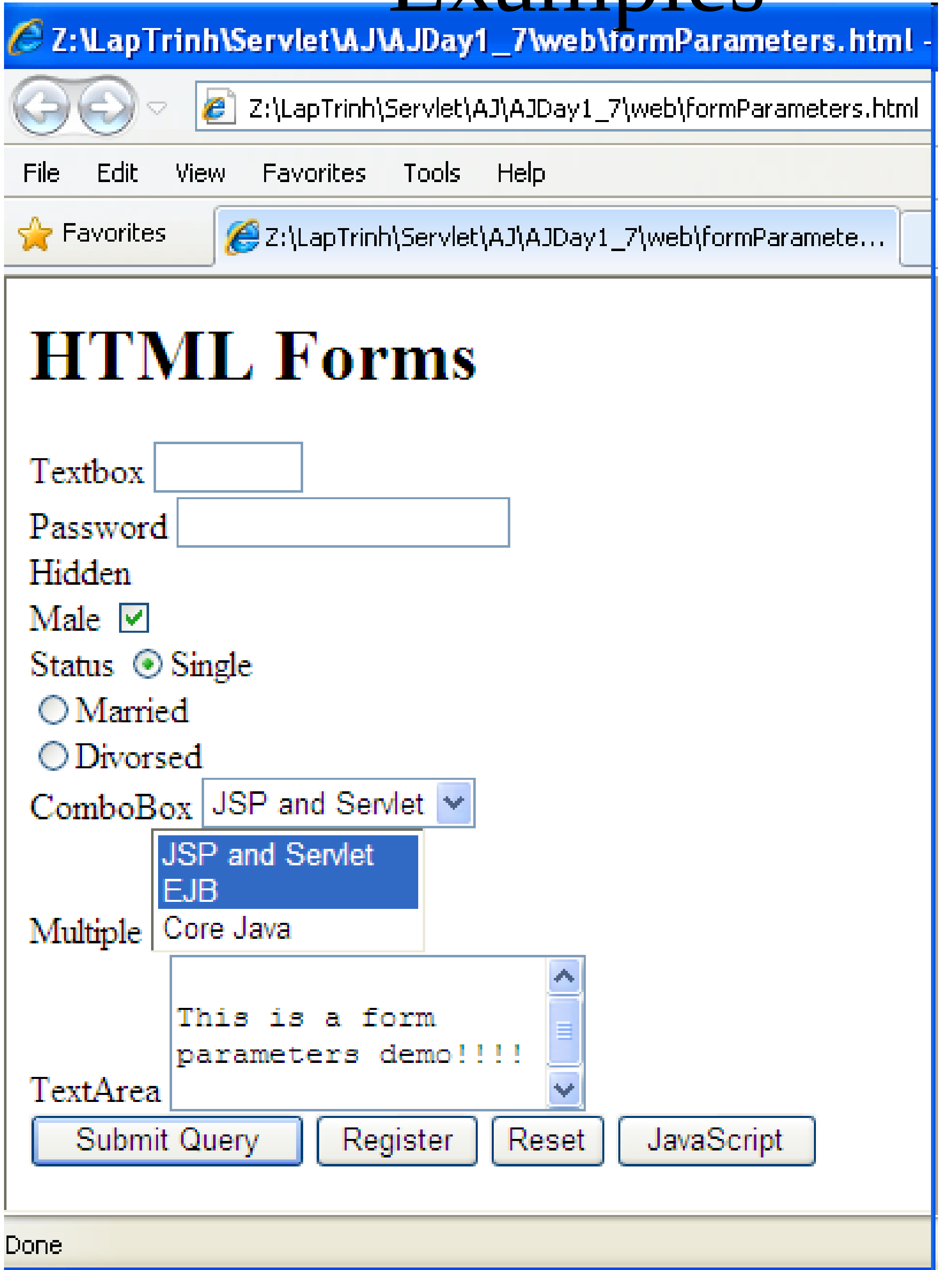
Form Parameters

Examples



```
    <body>
        <h1>HTML Forms</h1>
        <form action="index.html">
            Textbox <input type="text" name="txtText" value="" size="5" /><br/>
            Password <input type="password" name="txtPassword" value="" /><br/>
            Hidden <input type="hidden" name="txtHidden" value="" /><br/>
            Male <input type="checkbox" name="chkCheck" value="ON" checked="checked" /><br/>
            Status
            <input type="radio" name="rdoStatus" value="Single" checked="checked" />Single<br/>
            <input type="radio" name="rdoStatus" value="Married" />Married<br/>
            <input type="radio" name="rdoStatus" value="Divorsed" />Divorsed<br/>
            ComboBox <select name="txtCombo">
                <option value="Servlet">JSP and Servlet</option>
                <option value="EJB">EJB</option>
            </select><br/>
            Multiple <select name="txtList" multiple="multiple" size="3">
                <option value="Servlet" selected>JSP and Servlet</option>
                <option value="EJB" selected>EJB</option>
                <option value="Java">Core Java</option>
            </select><br/>
            TextArea <textarea name="txtArea" rows="4" cols="20">
                This is a form parameters demo!!!!
            </textarea><br/>
            <input type="submit" name="txtB" />
            <input type="submit" value="Register" name="action" />
            <input type="reset" name="txtB" />
            <input type="button" value="JavaScript" name="txtB" onclick="" />
        </form>
    </body>
</html>
```

Form Parameters Examples



Z:\LapTrinh\Servlet\AJ\AJDay1_7\web\formParameters.html -

Z:\LapTrinh\Servlet\AJ\AJDay1_7\web\formParameters.html
File Edit View Favorites Tools Help
Favorites
Z:\LapTrinh\Servlet\AJ\AJDay1_7\web\formParamete...

HTML Forms

Textbox

Password

Hidden

Male ☑

Status ● Single

○ Married

○ Divorsed

ComboBox JSP and Servlet

Multiple JSP and Servlet
EJB
Core Java

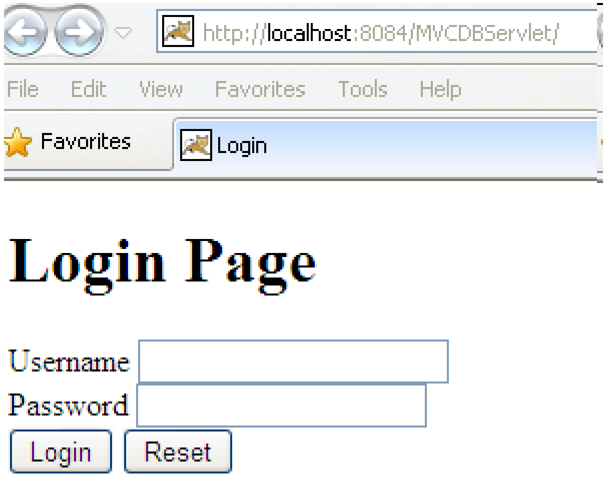
TextArea

```
This is a form
parameters demo!!!!
```

Submit Query
Register
Reset
JavaScript
Done

Build The Simple Web

Views

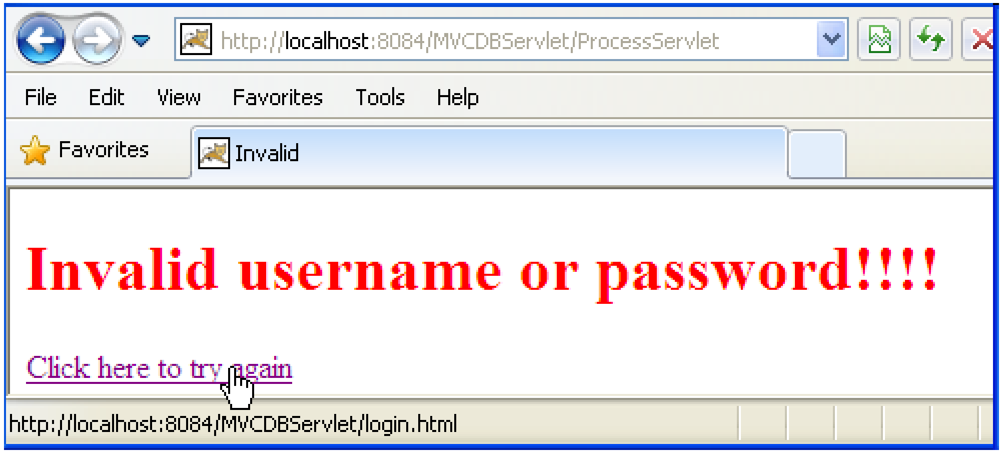


http://localhost:8084/MVCDBServlet/
File Edit View Favorites Tools Help
Favorites
Login

Login Page

Username

Password

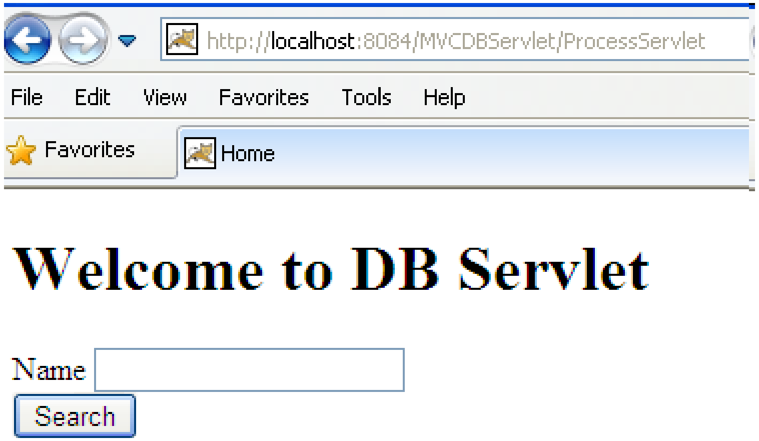


http://localhost:8084/MVCDBServlet/ProcessServlet
File Edit View Favorites Tools Help
Favorites
Invalid

Invalid username or password!!!!

Click here to try again

http://localhost:8084/MVCDBServlet/login.html



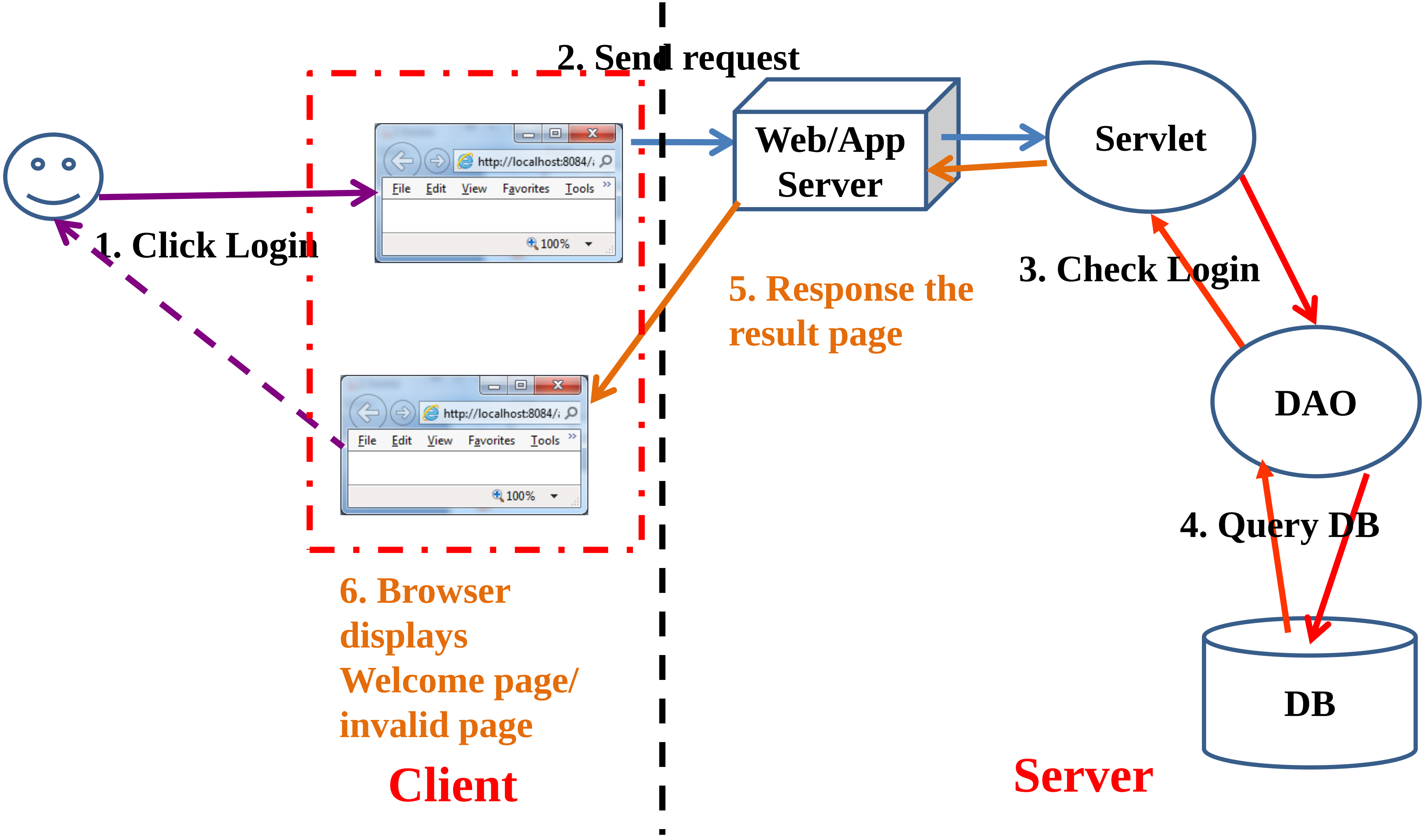
http://localhost:8084/MVCDBServlet/ProcessServlet
File Edit View Favorites Tools Help
Favorites
Home

Welcome to DB Servlet

Name

Build The Simple Web

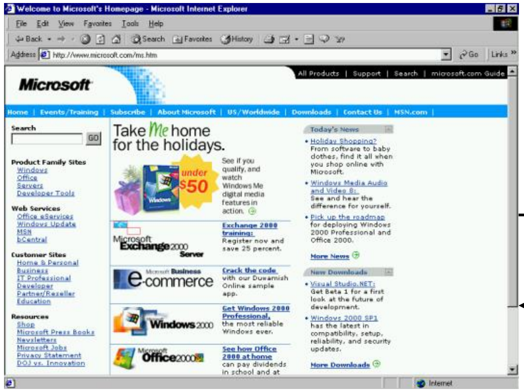
Interactive Server Model



HTTP Protocols

Overview

1. Convert http://microsoft.com/ to 192.168.54.3:80



Take Me home for the holidays.

under $50

See if you qualify, and watch Windows Me digital media features in action.

Microsoft Exchange 2000 Server

Exchange 2000 training: Register now and save 25 percent.

Microsoft Business e-commerce

Crack the code with our Duwamish Online sample app.

Windows 2000

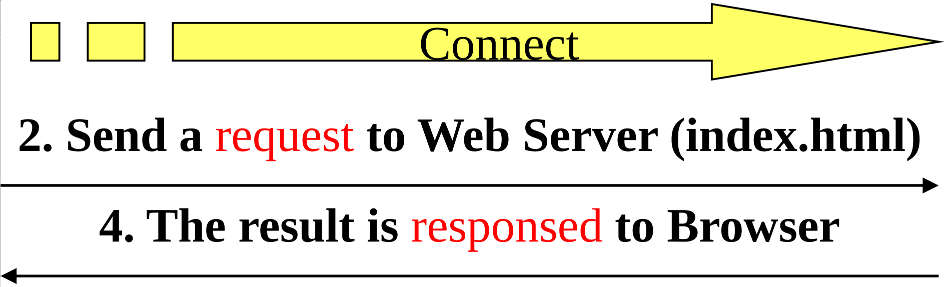
Get Windows 2000 Professional, the most reliable Windows ever.

Microsoft Office 2000

See how Office 2000 at home can pay dividends in school and at

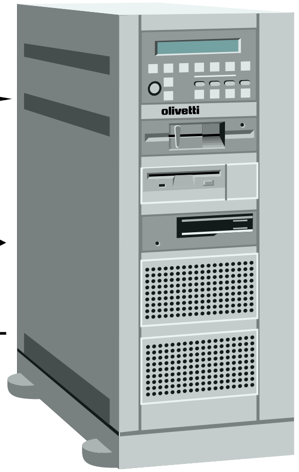
- Holiday Shopping? From software to baby clothes, find it all when you shop online with Microsoft.
- Windows Media Audio and Video 8: See and hear the difference for yourself.
- Pick up the roadmap for deploying Windows 2000 Professional and Office 2000.

- Visual Studio.NET: Get Beta 1 for a first look at the future of development.
- Windows 2000 SP1 has the latest in compatibility, setup, reliability, and security updates.



2. Send a request to Web Server (index.html)

4. The result is responsed to Browser



http://microsoft.com/index.html

192.168.54.3:80

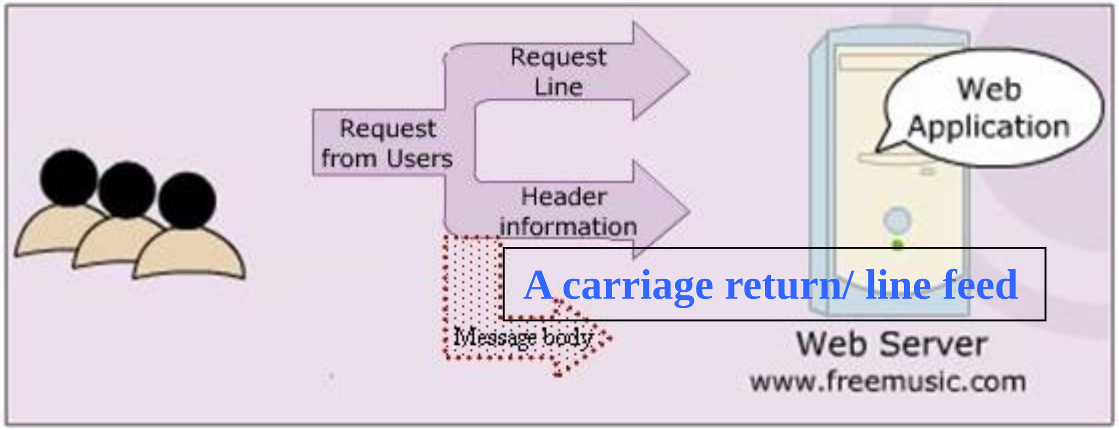
5. Web Browser views the result which contains a markup language

3. Web Server processes a request (connecting DB, calculating, call service …)

- **Request – Response** pairs
- **Stateless**
- Port **80** is default

HTTP Protocols

HTTP Requests

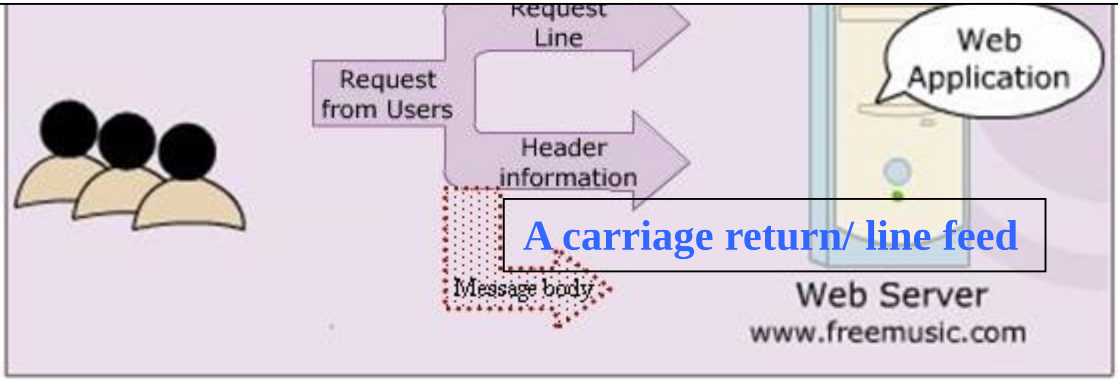


A carriage return/ line feed

HTTP Protocols

HTTP Requests

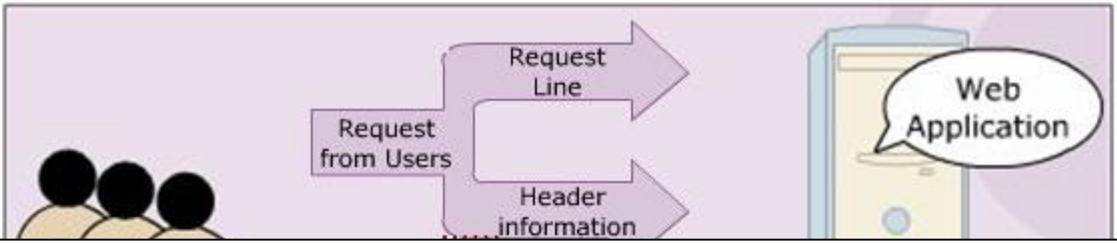
- **The HTTP method**
- **A pointer to the resource requested, in the form of a URI**
- **The version of HTTP protocol**
- Ex: **GET** /index.html HTTP/1.1



A carriage return/ line feed

HTTP Protocols

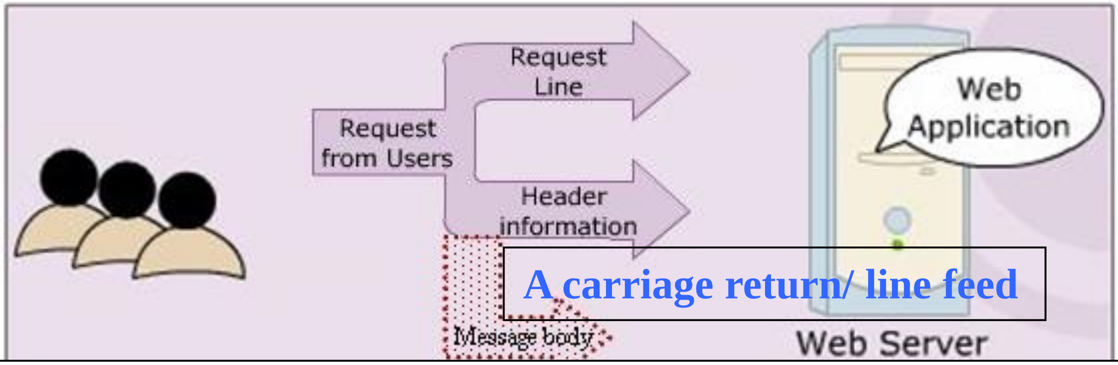
HTTP Requests



- Return the User-Agent (the **browser**) along **with** the **Accept header** in the form **name:value** (provides information on what media types the client can accept)
- **Ex**: User-Agent: Mozilla/4.0 (compatible: MSIE 4.0 : Windows 95) Accept : image/gif, image/jpeg, text/*, */*

HTTP Protocols

HTTP Requests

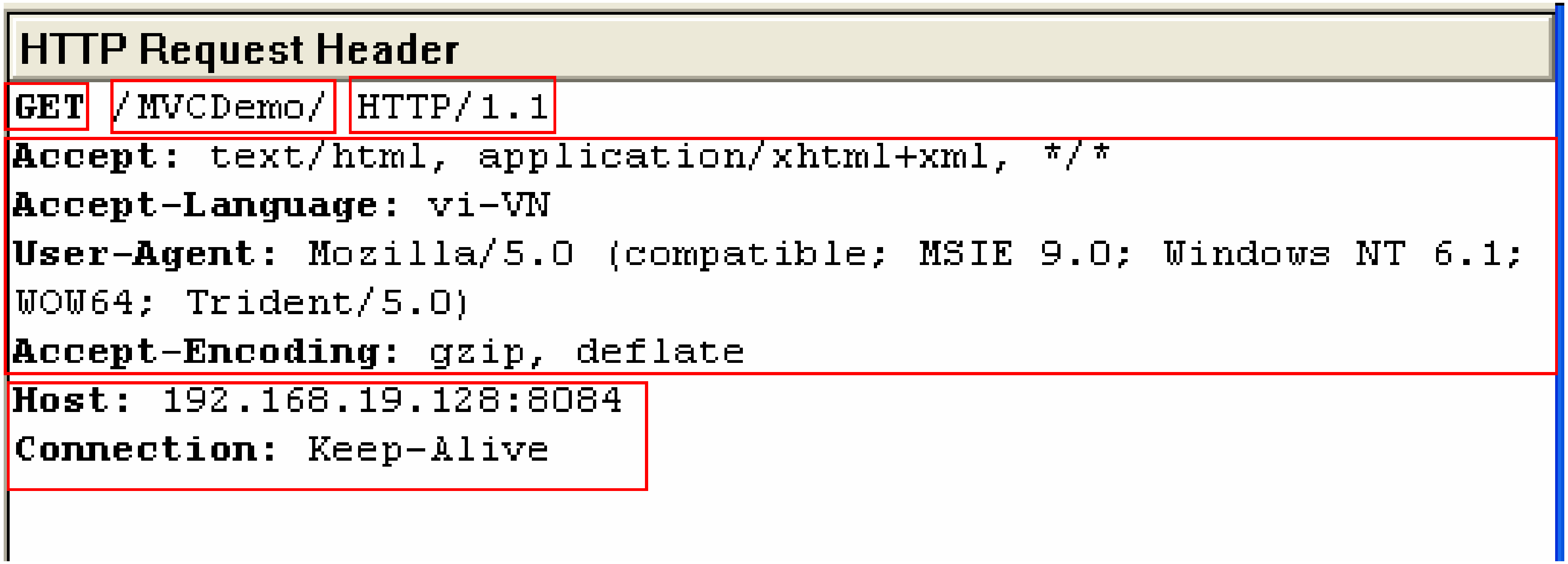


A carriage return/ line feed

- Contain pretty much any thing (a **set** of **parameters** and **values**, an **image** file intending to upload)

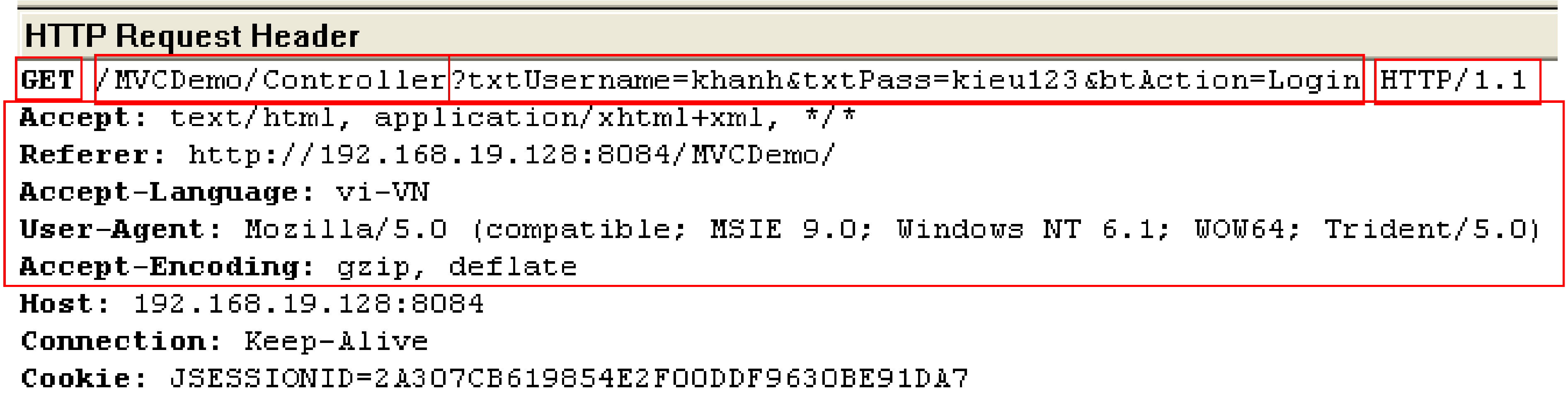
HTTP Protocols

HTTP Requests – Example



HTTP Request Header

```
GET /MVCDemo/ HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: vi-VN
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;
WOW64; Trident/5.0)
Accept-Encoding: gzip, deflate
Host: 192.168.19.128:8084
Connection: Keep-Alive
```

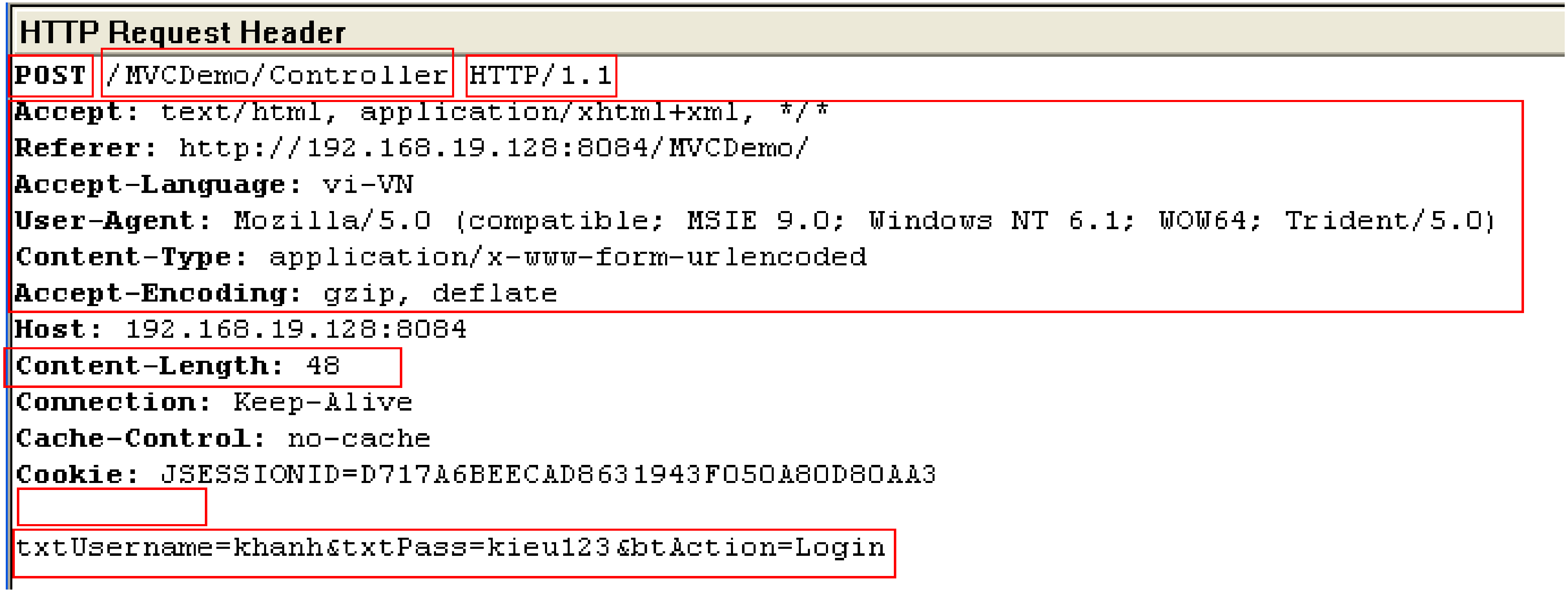


HTTP Request Header

```
GET /MVCDemo/Controller?txtUsername=khanh&txtPass=kieu123&btAction=Login HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://192.168.19.128:8084/MVCDemo/
Accept-Language: vi-VN
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Accept-Encoding: gzip, deflate
Host: 192.168.19.128:8084
Connection: Keep-Alive
Cookie: JSESSIONID=2A307CB619854E2F00DDF9630BE91DA7
```

HTTP Protocols

HTTP Requests – Example

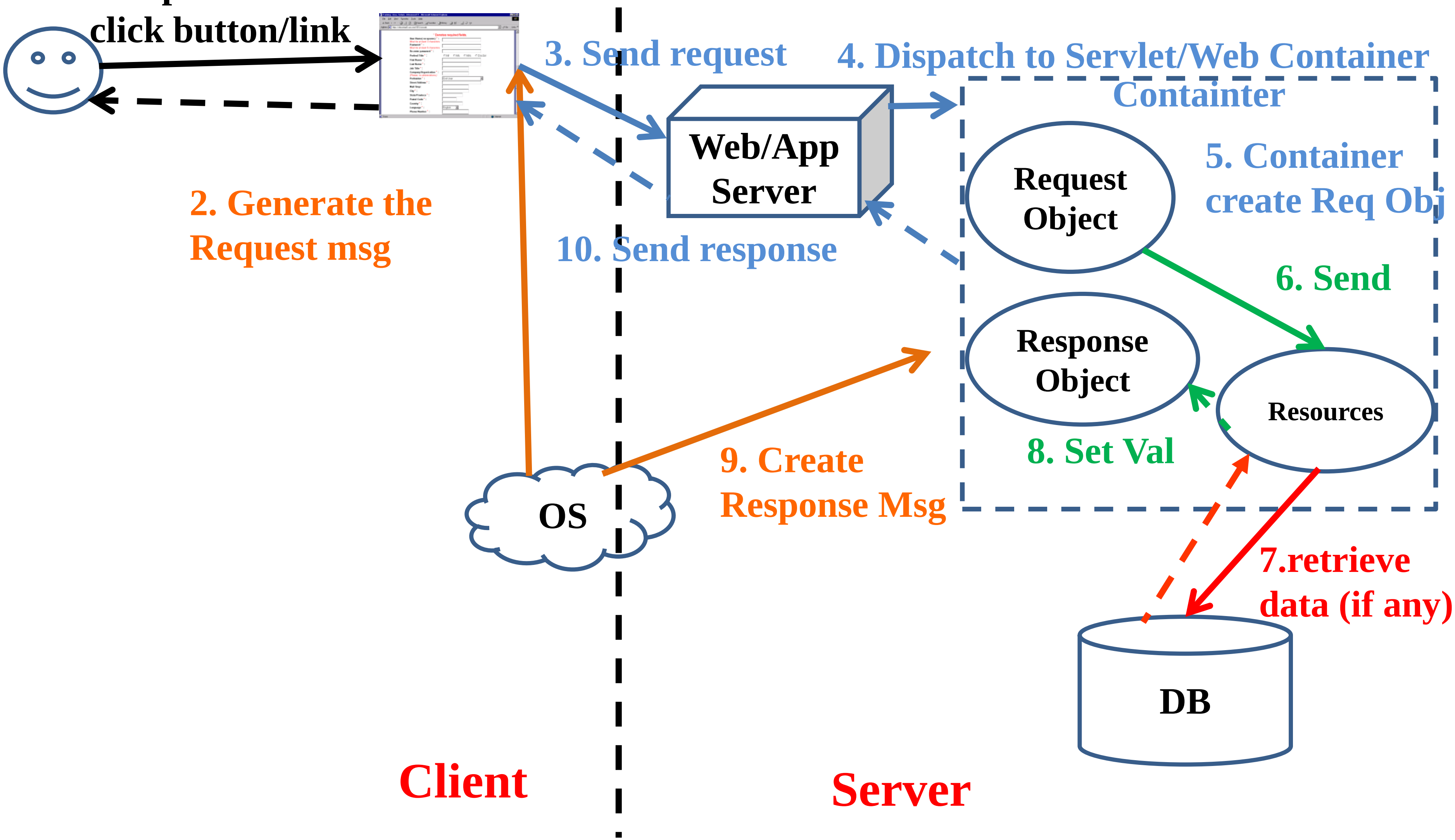


HTTP Request Header

```
POST /MVCDemo/Controller HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://192.168.19.128:8084/MVCDemo/
Accept-Language: vi-VN
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: 192.168.19.128:8084
Content-Length: 48
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: JSESSIONID=D717A6BEECAD8631943F050A80D80AA3

txtUsername=khanh&txtPass=kieu123&btAction=Login
```

The Servlet Model

Request Objects



1. Input and click button/link

2. Generate the Request msg

3. Send request

4. Dispatch to Servlet/Web Container

Web/App Server

Containter

5. Container create Req Obj

Request Object

6. Send

Resources

7.retrieve data (if any)

DB

8. Set Val

Response Object

9. Create Response Msg

OS

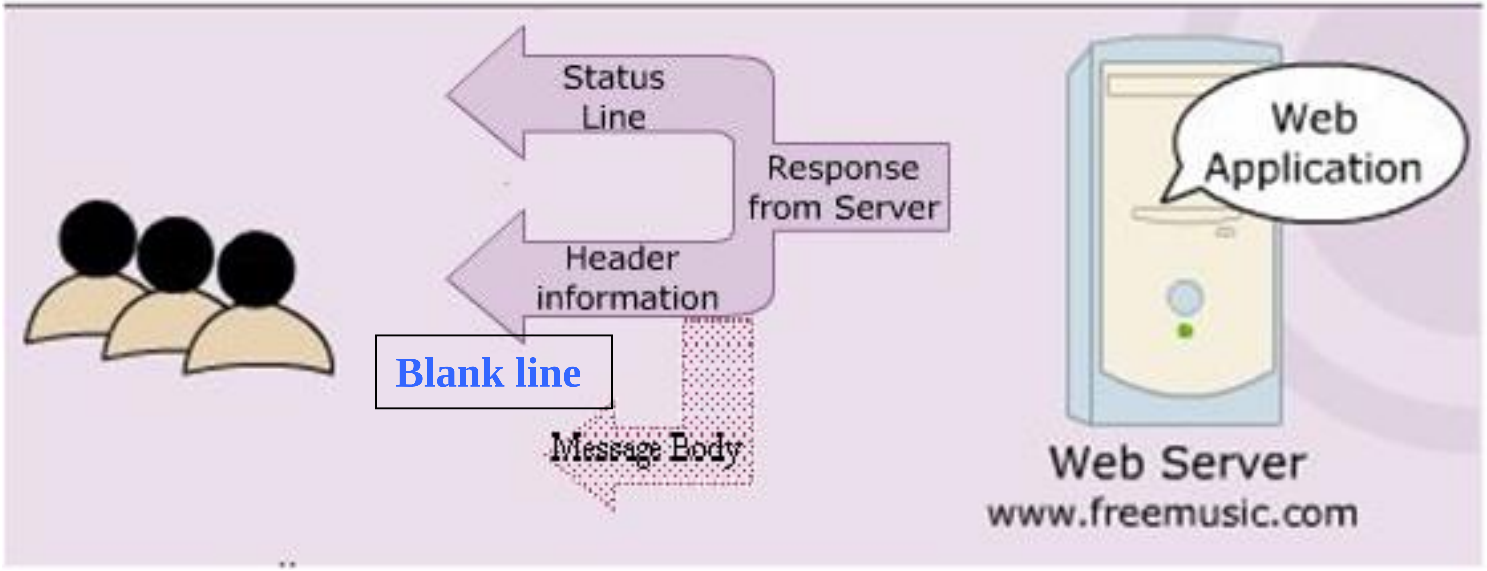
10. Send response

Client

Server

HTTP Protocols

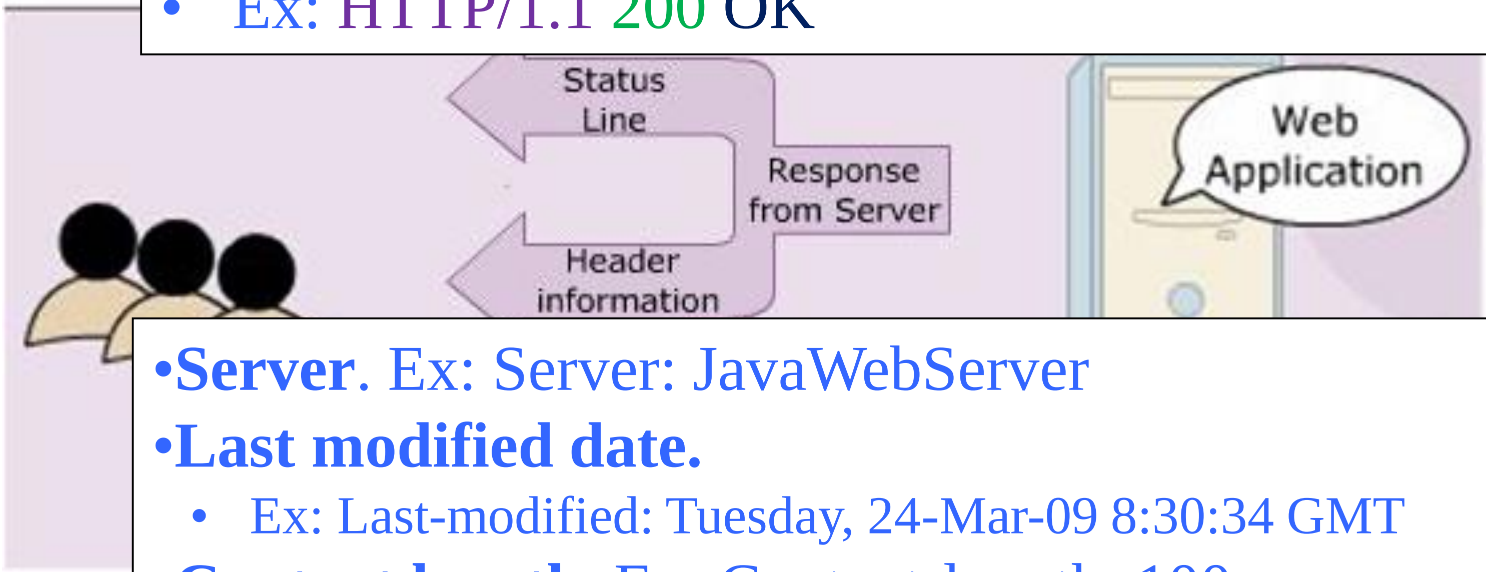
HTTP Responses



HTTP Protocols

HTTP Responses

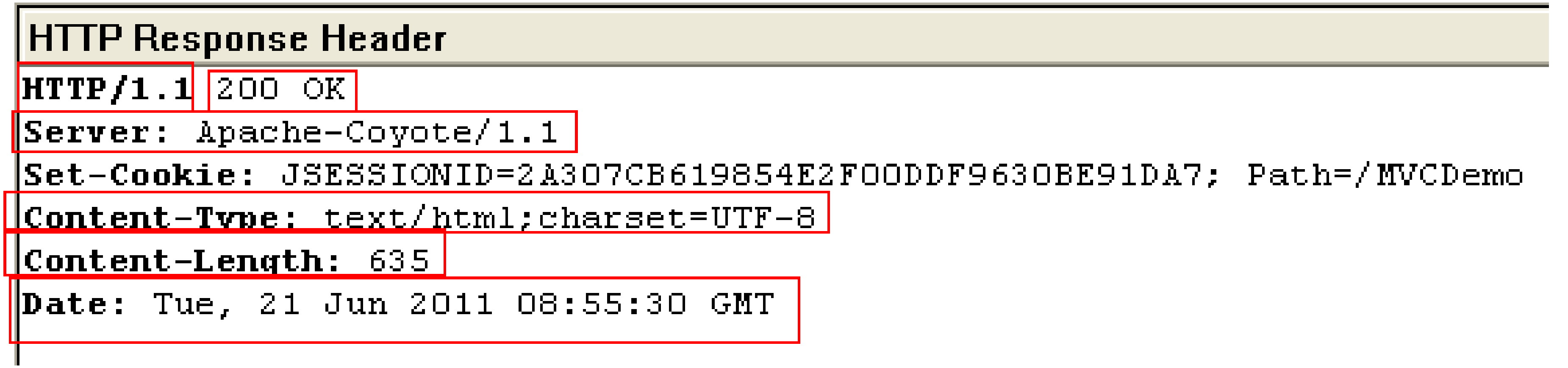
- **Indicates status of request process (HTTP version, response code, status)**
- Ex: HTTP/1.1 200 OK



- **Server**. Ex: Server: JavaWebServer
- **Last modified date.**
  - Ex: Last-modified: Tuesday, 24-Mar-09 8:30:34 GMT
- **Content length**. Ex: Content-length: 100
- **Content type**. Ex: Content-type: text/plain

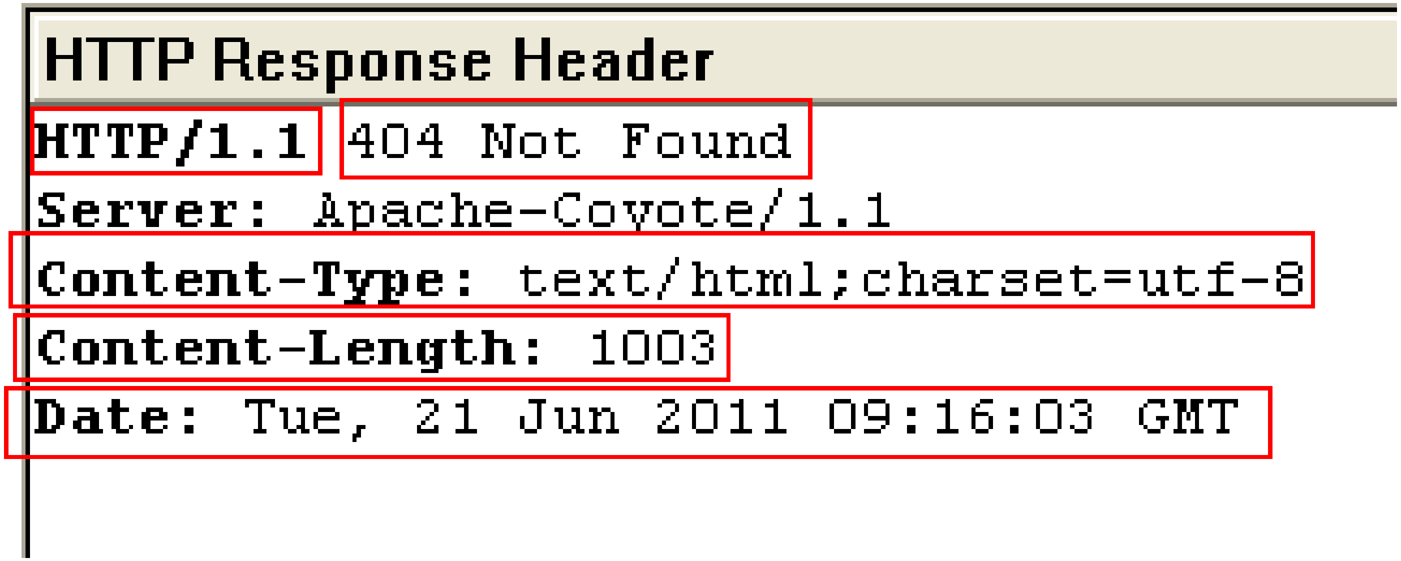
HTTP Protocols

HTTP Responses – Example



HTTP Response Header

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=2A307CB619854E2F00DDF9630BE91DA7; Path=/MVCDemo
Content-Type: text/html;charset=UTF-8
Content-Length: 635
Date: Tue, 21 Jun 2011 08:55:30 GMT
```

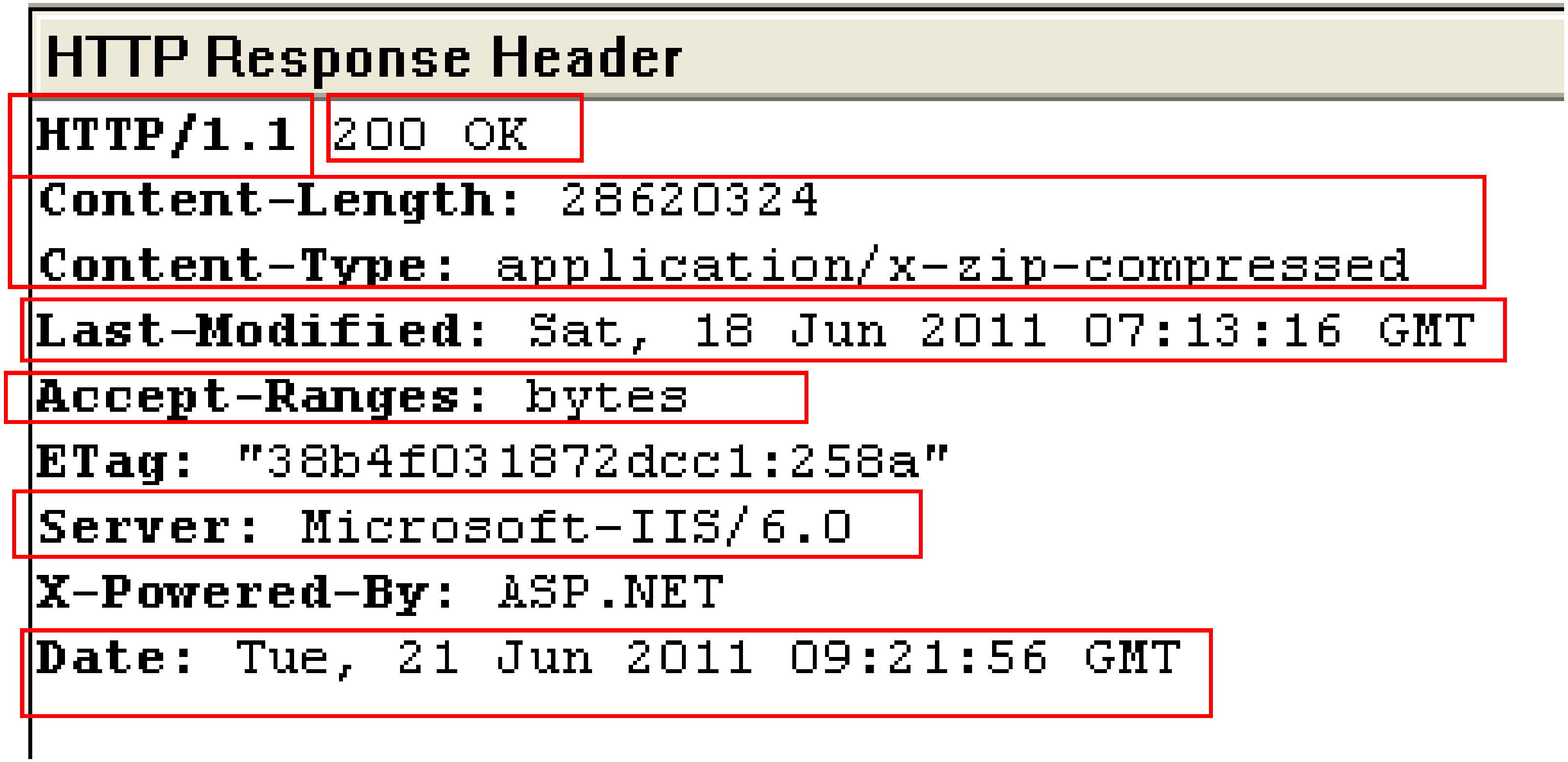


HTTP Response Header

```
HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
Content-Length: 1003
Date: Tue, 21 Jun 2011 09:16:03 GMT
```

HTTP Protocols

HTTP Responses – Example



HTTP Response Header

```
HTTP/1.1 200 OK
Content-Length: 28620324
Content-Type: application/x-zip-compressed
Last-Modified: Sat, 18 Jun 2011 07:13:16 GMT
Accept-Ranges: bytes
ETag: "38b4f031872dcc1:258a"
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Date: Tue, 21 Jun 2011 09:21:56 GMT
```

HTTP Protocols

Some commonly Status codes

Code	Associated Message	Meaning
101	Switching Protocols	- Server will **comply** with **Upgrade header** and **change** to **different protocol**. (New in HTTP 1.1)
200	OK	- **Everything** is **fine**; document follow - **Default** for servlets
201	Created	- Server **created** a **document** - The Location header indicates its URL
203	Non-Authoritative Information	- Document is being **returned normally**, **but** some of the **response headers** might be **incorrect** since a **document copy is being used**.
204	No Content	- Browser should **keep displaying previous** document
301	MovedPermanently	- **Document is moved to** a **separate location** as mentioned in the URL. - The page is **redirected to the mentioned URL**, to find the document
302	Found	- Temporary **replacement of file** from one location to the other as specified

HTTP Protocols

Some commonly Status codes

Status code	Associated Message	Meaning
400	Bad Request	- The **request placed** is **syntactically incorrect**
401	Unauthorized	- Authorization **not given to access** a password protected page
403	Permission denied	- **Authentication but authorization not given** to access protected resource
404	Not Found	- **Resource not found** in the specified address
408	Request Timeout	- **Time taken** by **client** is **very long to send** the **request** (only available in HTTP 1.1)
500	Internal Server Error	- Server is unable to locate the requested file. The servlet has been **deleted or crashed or** had been moved to a new location with out informing
503		- Indicates that the **HTTP server is temporarily overloaded,** and unable to handle the request
…	…	-…

HTTP Protocols

HTTP Methods – Basic

- **GET**
  - Is the method commonly used to **request a resource/ get information** *(access static resource such as HTML doc and images or retrieve dynamic information such as query parameters)* **from server**
  - The **restricted length of query string**, that is introduced by the question mark “?”
  - Is **trigger** by
    - **Typing** into the address line of the browser and pressing GO
    - **Clicking** on a **link** in a web page
    - **Pressing** the **submit button** in an HTML **form** with **GET method**
- **POST**
  - **Sends** data of **unlimited length** to the web server.
  - Is the method commonly used for passing user input/ sending information to the server (***access dynamic resources*** *and* ***enable secure data*** *in HTTP request because the request parameters are passed in the body of request*)
  - **No limit** and **cannot be booked mark** or emailed

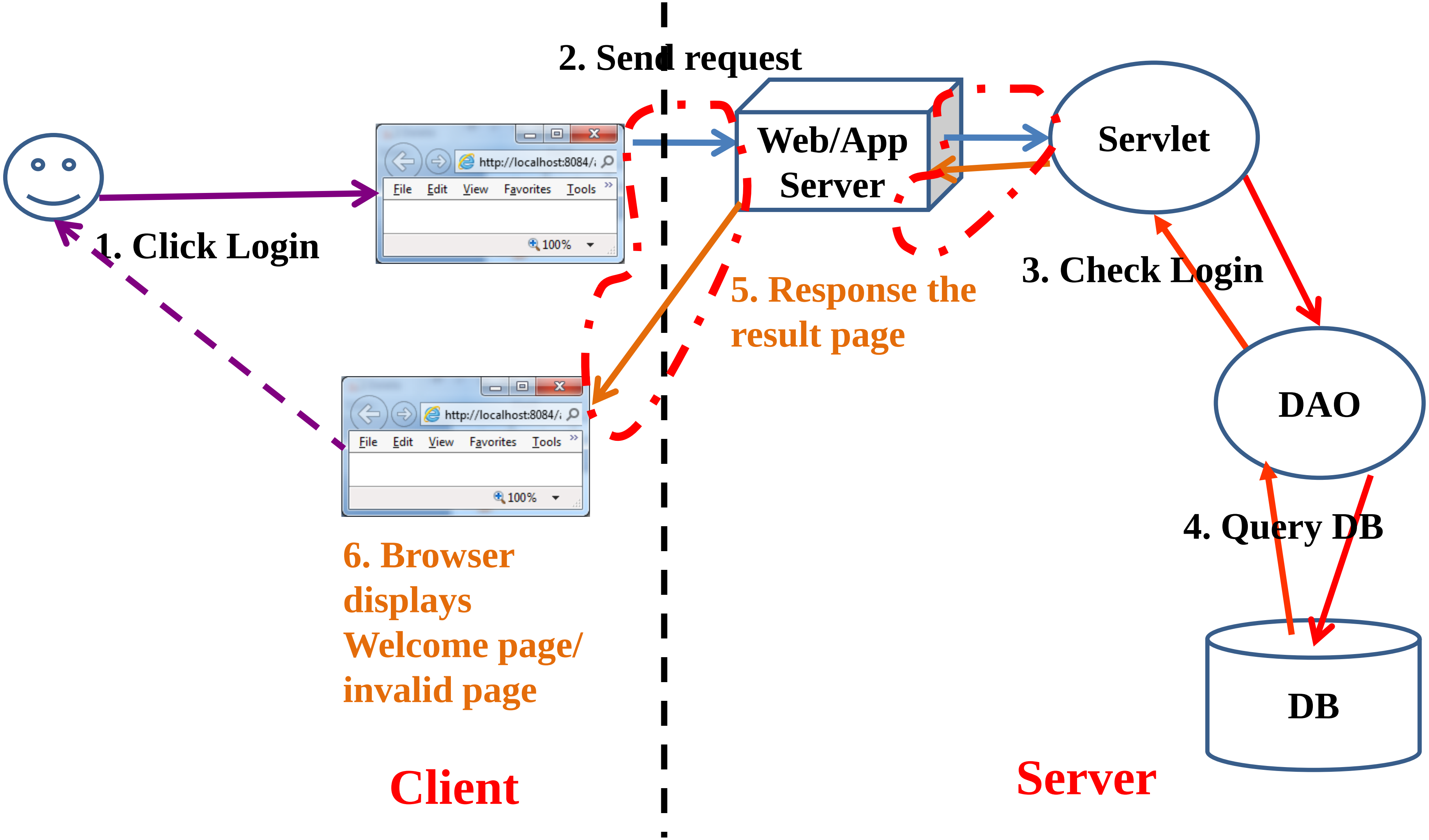
HTTP Protocols

HTTP Methods – Extends

- **HEAD**
  - **Returns** the **headers** identified by the **request** URL.
  - Is identical to the GET method but it doesn't return a message body
  - Is an economical way of checking that a resource is valid and accessible
- **OPTIONS**
  - **Returns** the **HTTP methods** the server supports.
- **PUT**
  - **Requests** the server to **store** the **data** enclosed in the HTTP message body **at a location provided in the request URL.**
- **DELETE**
  - **Requests** the server to **delete** the **resource identified** by the request URL.
- **TRACE**
  - Is **used** for **debugging** and **testing** the **request** sent to the server. It is **useful** when the **request** sent to the **server reaches** through the proxies.
- **Idempotency** and **Safety**
  - GET, TRACE, OPTIONS, and HEAD

Build The Simple Web

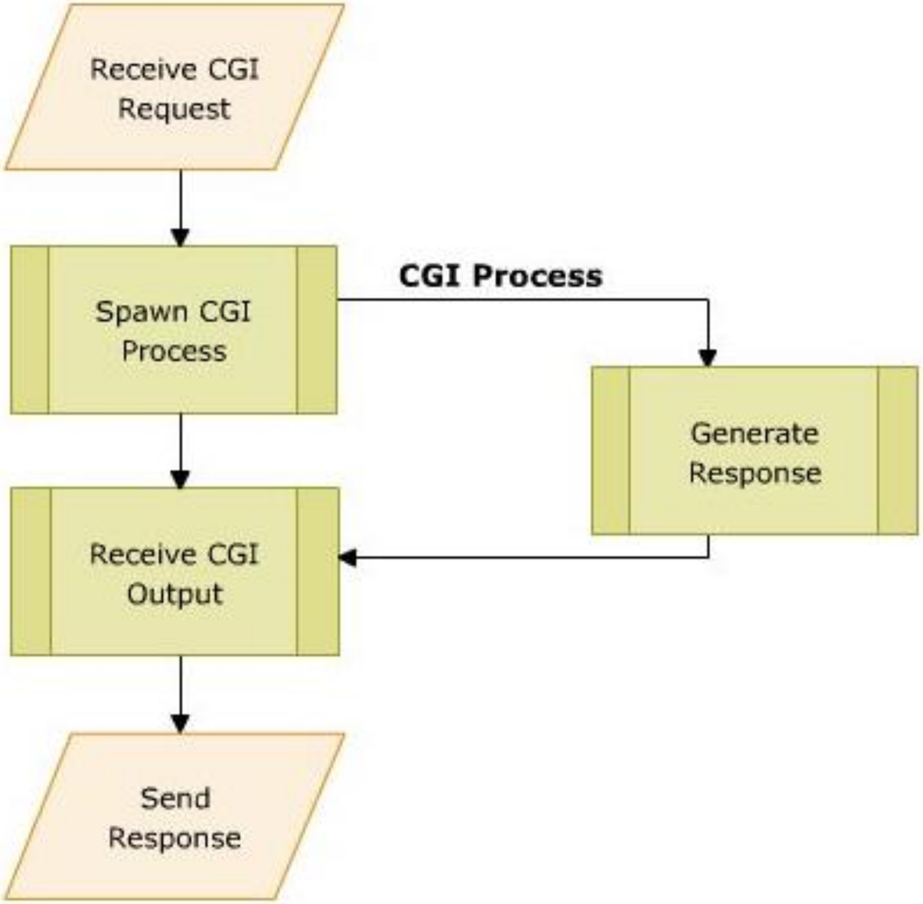
Interactive Server Model



The Servlet Model

Common Gateway Interface (CGI)

- A **small program (*.exe)** is **written** in **languages** such as **C/C++, Perl**, ... for the gateway programs.
- Used in complex applications, such as **Web pages**
- A set of standards followed to **interface applications form client side** to a Web Server
- Enables the Web server to send information to other files and Web browsers
- Helps to **process the inputs** to the form on the Web page
- Enables to **obtain information** and use it on the server machine (server side)
- When the **Browser sends request** to server, **CGI instantaties** to **receive and process**.

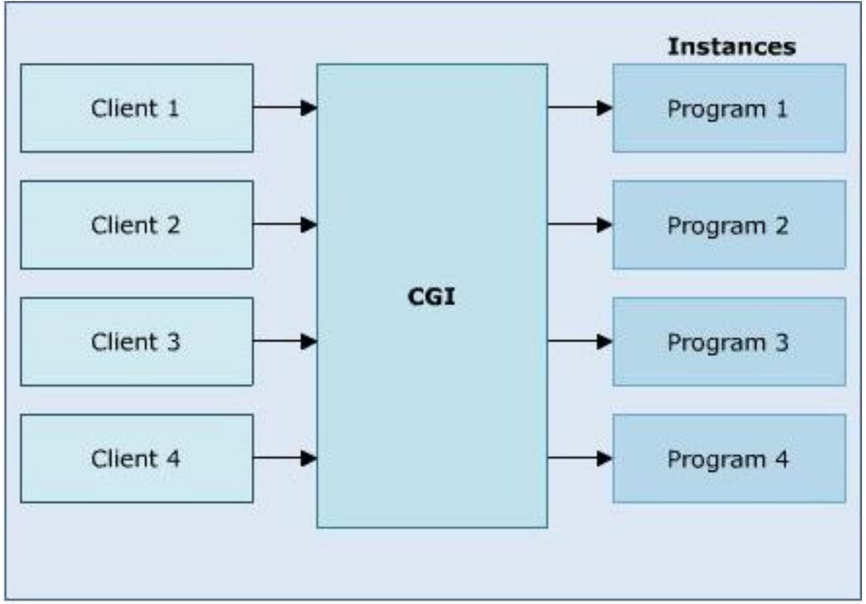


Server Process for running CGI

The Servlet Model

Common Gateway Interface (CGI)

- Disadvantages
  - **Reduced efficiency**



CGI process

The Servlet Model

Common Gateway Interface (CGI)

- Disadvantages
  - **Reduced efficiency**
  - **Reloading Perl interpreter**
    - The widely accepted platform for writing CGI script is Perl. Each time the server receives are quest, the Perl interpreter needs to be reloaded.
  - **Interactive**: not suitable for graphical or highly interactive programs
  - **Time consuming and more memory consumed**
  - **Debugging**: error detection is difficult
  - **Not support Session**

The Servlet Model

Servlets

- Are **Java classes** that **dynamically process** HTTP **requests** and **construct responses**
- Are **Java codes** that are used to **add dynamic** content to Web server.
- There is **only a single instance** of Servlet created on the Web server.
- To **service multiple clients' request**, the Web server **creates multiple threads** for the same Servlet instance (**Overcome CGI's consumed more memory**)
- Gets **auto refreshed** on receiving a request each time
- A Servlet's **initializing code** is used **only** for initializing **in the 1st time**
- **Merits**
  - Enhanced efficiency (initializing only once, auto refresh)
  - Ease to use (using Java combining HTML)
  - Powerful (using Java)
  - Portable
  - Safe and cheap
- **Demerits**
  - **Low-level HTML documentation** (Static well-formed-ness is not maintained)
  - **Unclear-session management** (flow of control within the codes is very unclear)

The Servlet Model

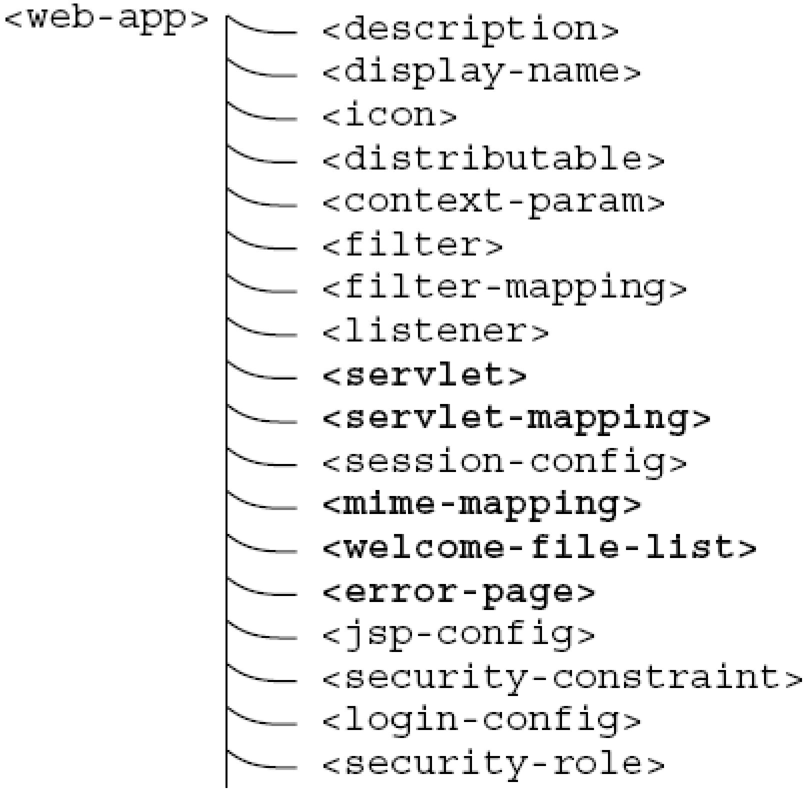
Servlets

- How to server **detecting** the **servlets** (difference from Java class), then, **initializing in the 1st time?**
  - **Web deployment descriptors (web.xml)**
  - **Annotations**

The Servlet Model

The Deployment Descriptor

- The Web Deployment Descriptor file **describes all** of **Web components**
- It is an **XML file**. Given that the name is **web.xml.**



<web-app>

<description>

<display-name>

<icon>

<distributable>

<context-param>

<filter>

<filter-mapping>

<listener>

<servlet>

<servlet-mapping>

<session-config>

<mime-mapping>

<welcome-file-list>

<error-page>

<jsp-config>

<security-constraint>

<login-config>

<security-role>

The Servlet Model

The Deployment Descriptor – web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <servlet>
        <servlet-name>servlet name</servlet-name>
        <servlet-class>package.classname</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>servlet name</servlet-name>
        <url-pattern>/context Path/root</url-pattern>
    </servlet-mapping>
    <session-config>
      <session-timeout>30</session-timeout>
     </session-config>
    <welcome-file-list>
        <welcome-file>default page to show</welcome-file>
    </welcome-file-list></web-app>
```

Servlets Declaration is same as **package.classname servlet_name;**

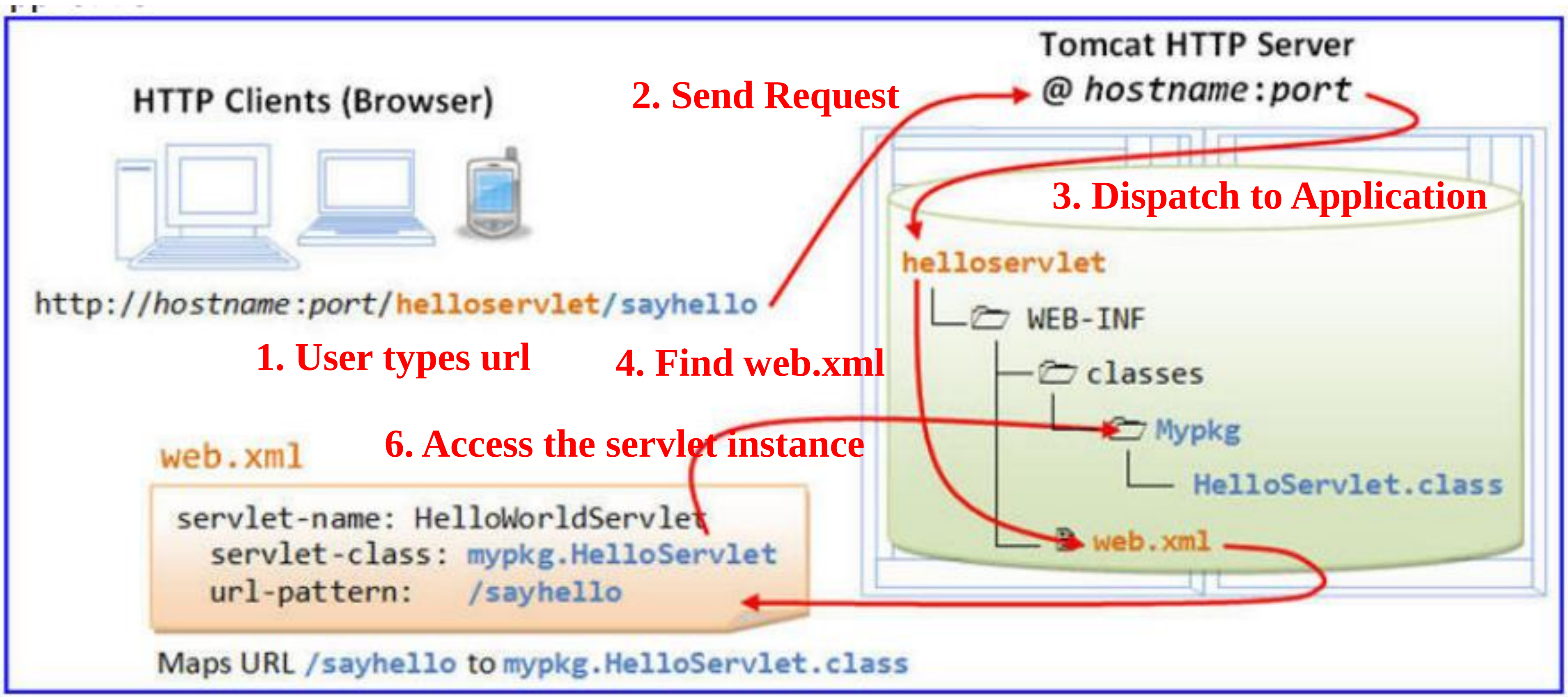
Define the access path to the servlet**;**

The Servlet Model
The Deployment Descriptor – Example

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <servlet>
            <servlet-name>HelloServlet</servlet-name>
            <servlet-class>servlet.sample.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
            <servlet-name>HelloServlet</servlet-name>
            <url-pattern>/HelloServlet</url-pattern>
    </servlet-mapping>
    <session-config>
      <session-timeout>30</session-timeout>
     </session-config>
    <welcome-file-list>
            <welcome-file>HelloServlet</welcome-file>
    </welcome-file-list></web-app>
```

The Servlet Model

The Deployment Descriptor – Example



```
servlet-name: HelloWorldServlet
   servlet-class: mypkg.HelloServlet
   url-pattern:    /sayhello
```

Maps URL /sayhello to mypkg.HelloServlet.class

1. User types url

2. Send Request

3. Dispatch to Application

4. Find web.xml

5. Look up the servlet class from mapping to find the servlet instance web.xml

6. Access the servlet instance

The Servlet Model

Annotations

- Are one of the **major advancement** from Java EE 5.0 that makes the standard **web.xml deployment descriptors** files **optional**
  - To **avoid writing** such kind of **unnecessary codes**, annotations are used
- Can be defined as **metadata information** that can be **attached** to an element **within the code** to characterize it
  - Simplifies the **developer's work** to a great extent by significantly **reducing** the **amount of code** to be **written** by moving the metadata information into the source code itself
- Are **never executed and processed** when the code containing it are **compiled or interpreted by compilers, deployment tools**, and so on
- An annotation type takes **an 'at (@)' sign**, followed by the interface keyword and the annotation name

The Servlet Model

Annotations – Servlets

- The javax.servlet.**annotation** package provides annotations to declare Servlets by specifying metadata information in the Servlet class

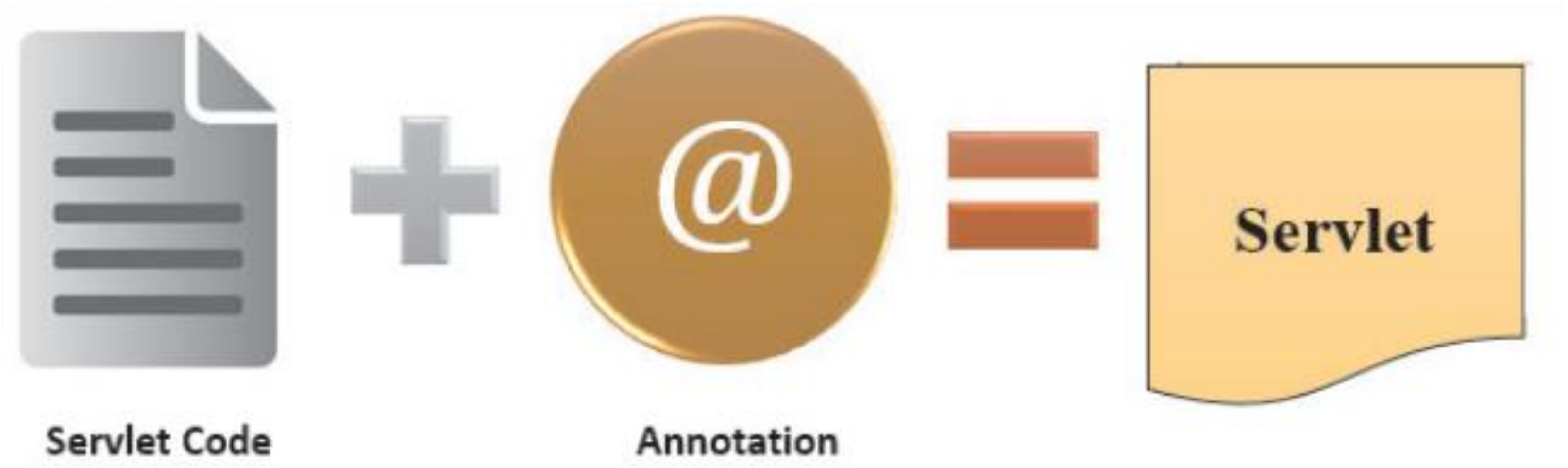


Figure 4.9: Servlet with Annotations, Web Component Development Using Java, Aptech World Wide

The Servlet Model

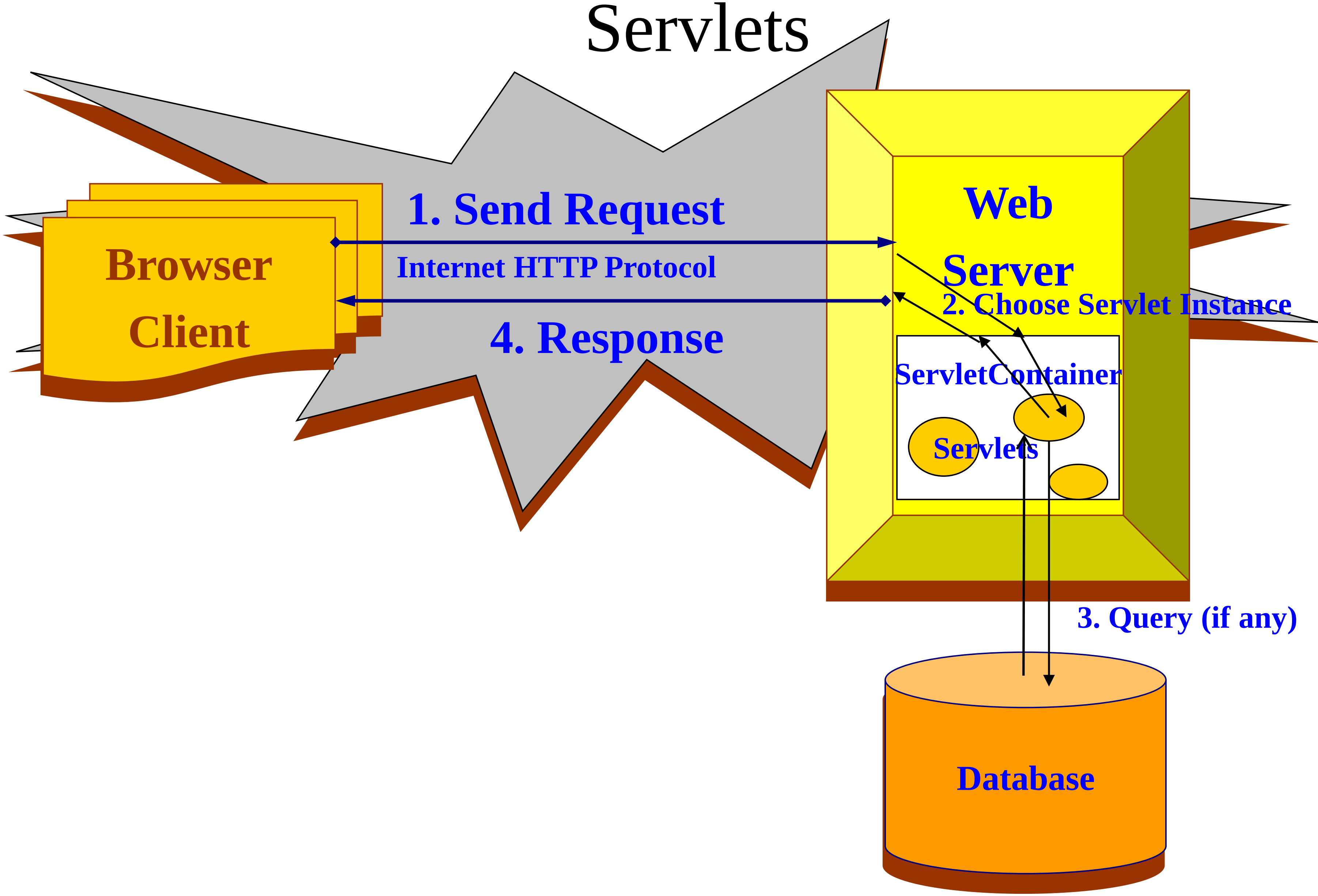
Annotations – Servlets

- **WebServlet**
  - Is used to provide the **mapping information of the Servlet**.
  - Is processed by the servlet container at the time of the **deployment**.

Attributes	Descriptions
name	Specifies the Servlet name. This attribute is optional.
urlPatterns	An array of url patterns use for accessing the Servlet, this attribute is required and should register one url pattern
initParams	An array of @WebInitParam, that can be used to pass servlet configuration parameters. This attribute is optional.
…	

The Servlet Model

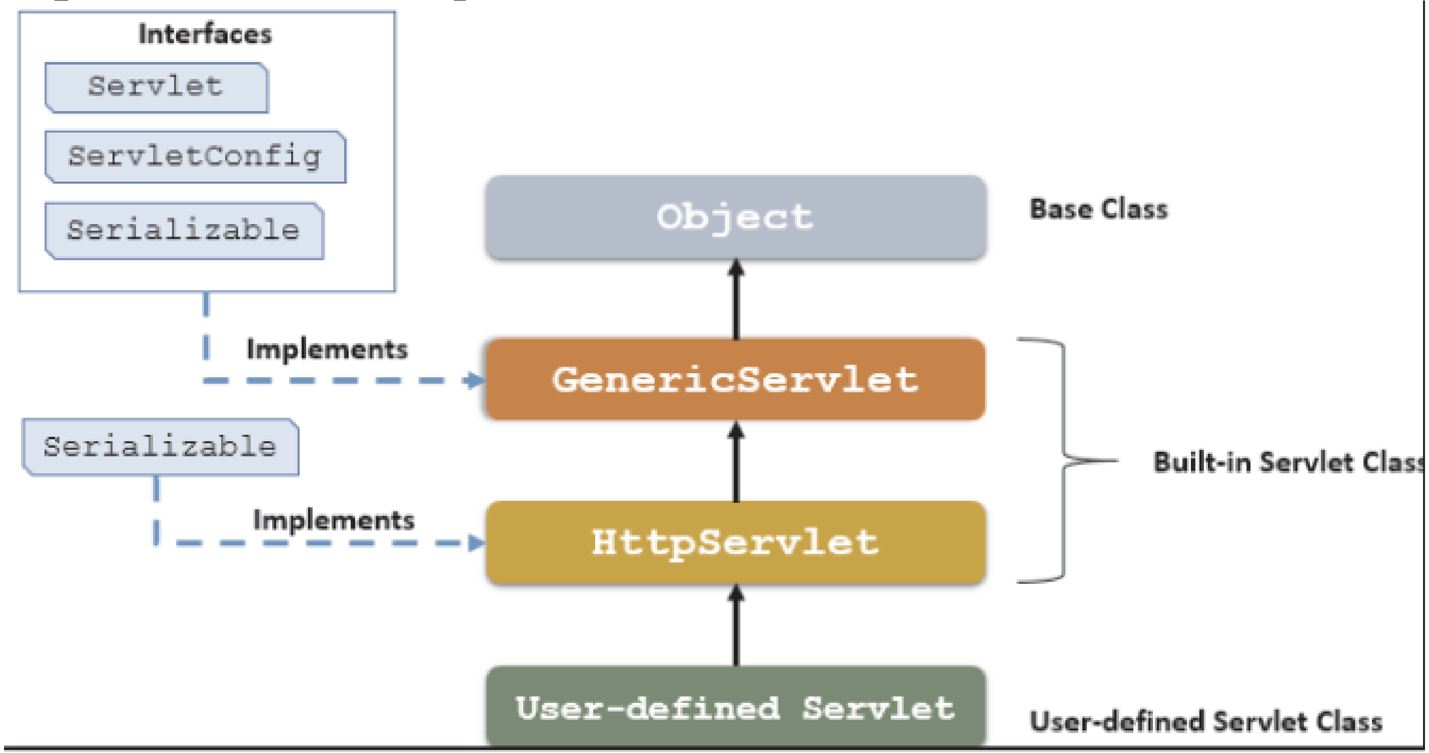
Servlets



The Servlet Model

Architecture of the Servlet packages

- The ***javax.servlet*** package provides interfaces and classes for writing servlets
  - The important interface is **javax.servlet.Servlet**
- When a servlet accepts a call from a client, it receives two objects:
  - **ServletRequest**, which encapsulates the communication from the client to the server.
  - **ServletResponse**, which encapsulates the communication from the servlet to the client.



The Servlet Model
GenericServlet class

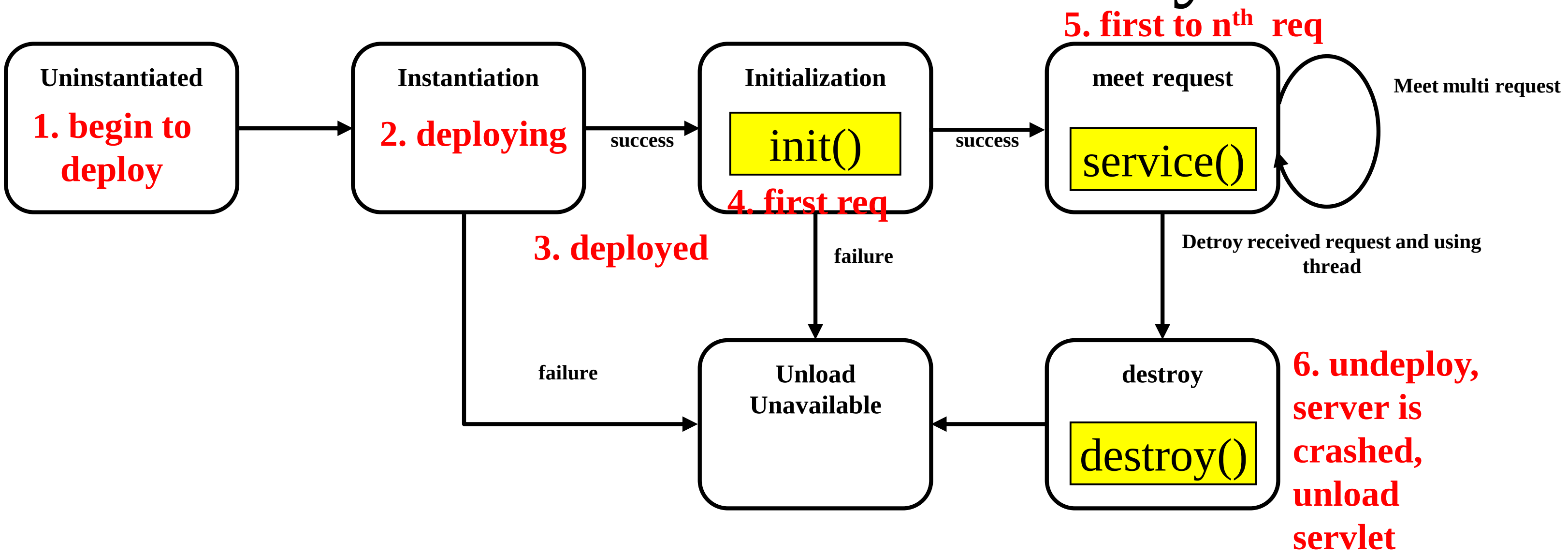
- Defines a **servlet** that **is not protocol dependent**
- **Implements** the **Servlet**, the **ServletConfig**, and the **java.io.Serializable** interfaces
- **Retrieves** the **configuration information** by implementing the ServletObject
- Some methods

Methods	Descriptions
init	- **public void init() throws ServletException** - Initializes the servlet
service	- **public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, IOException** - Called by the container to respond to a servlet request
destroy	- **public void destroy():** cleaning the servlet

Servlet Life Cycle defined in Generic

The Servlet Model

The Servlet Life Cycle

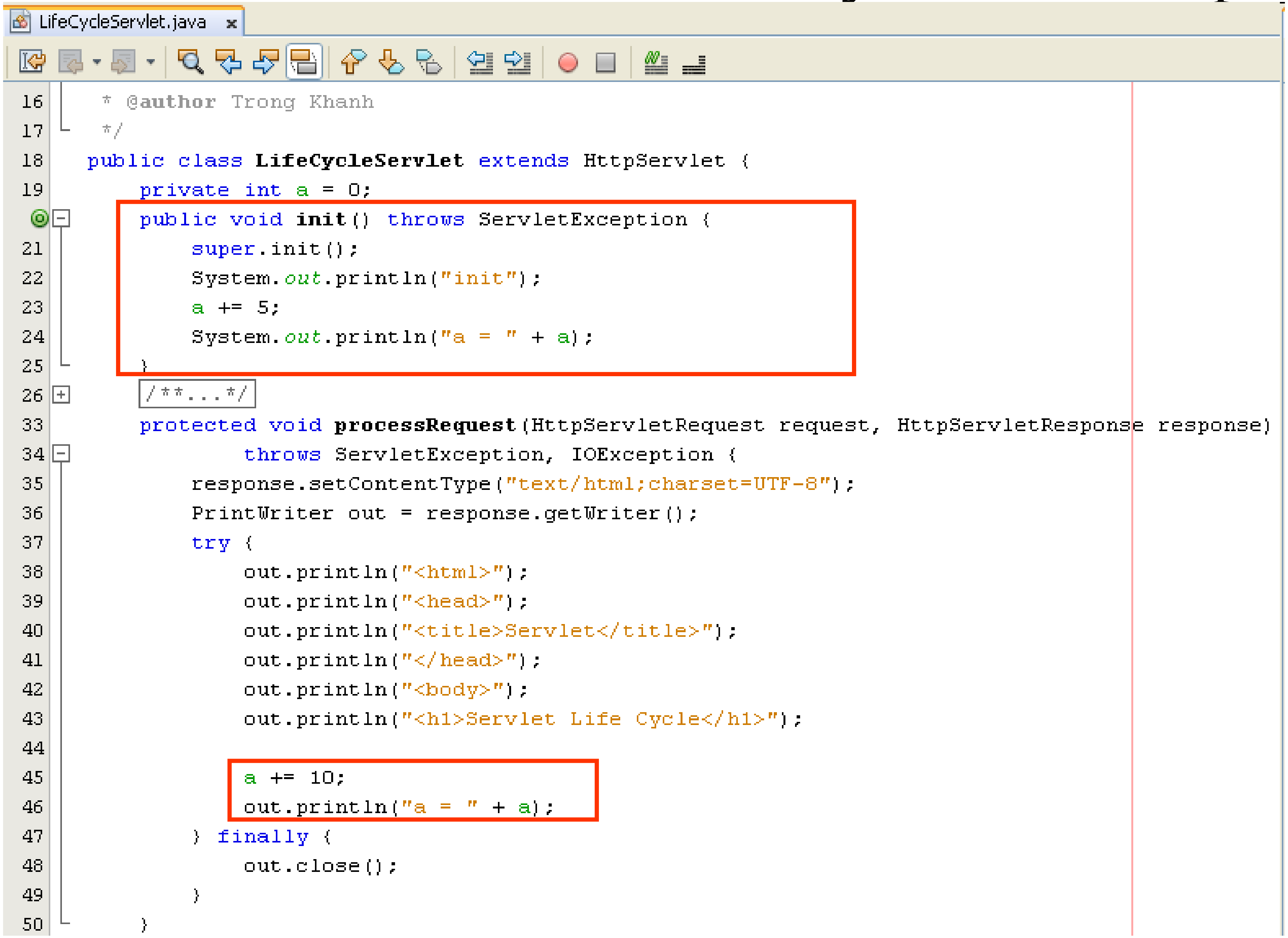


The life cycle is defined by

- **init()** – called only one by the server in the first request
- **service()** – process the client’s request, dispatch to doXXX() methods
- **destroy()** – called after all requests have been processed or a server-specific number of seconds have passed

The Servlet Model

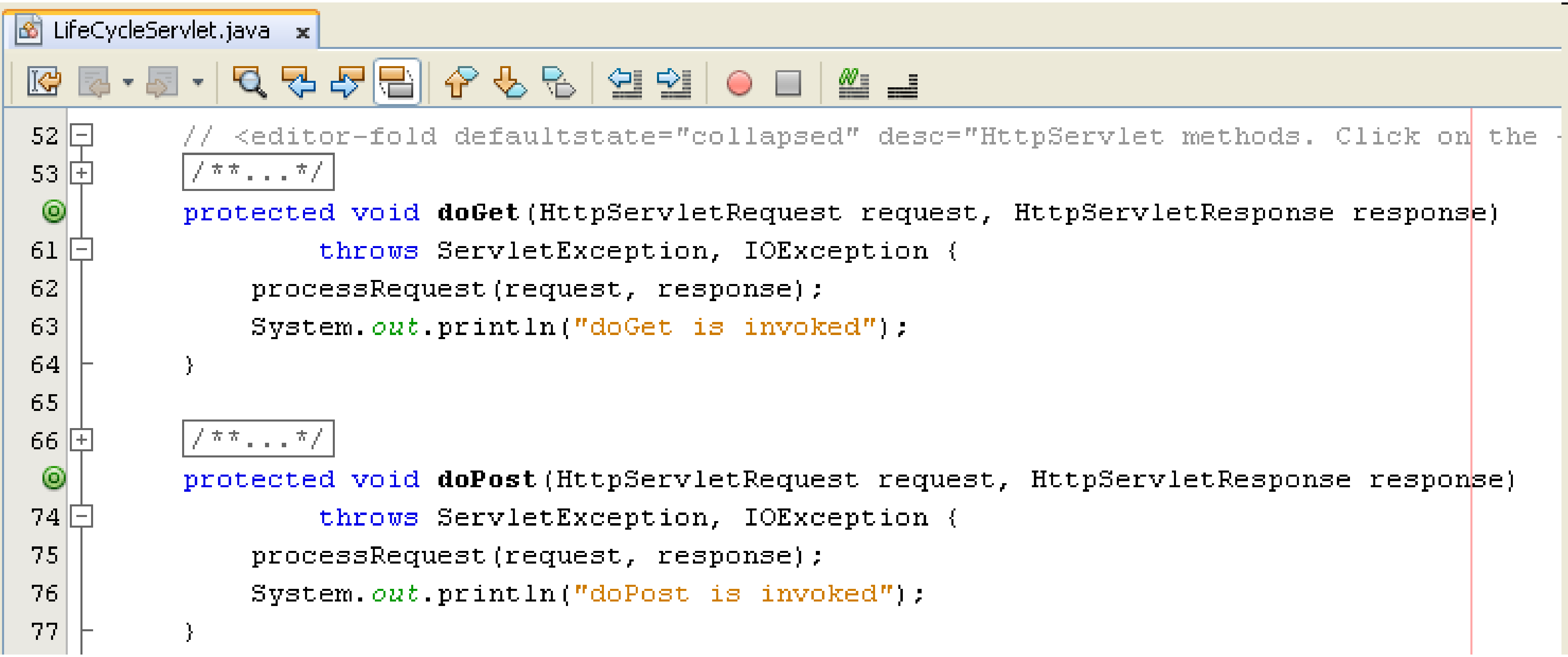
The Servlet Life Cycle – Example



```
 * @author Trong Khanh
 */
public class LifeCycleServlet extends HttpServlet {
    private int a = 0;
    public void init() throws ServletException {
        super.init();
        System.out.println("init");
        a += 5;
        System.out.println("a = " + a);
    }
    /**...*/
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet Life Cycle</h1>");

            a += 10;
            out.println("a = " + a);
        } finally {
            out.close();
        }
    }
```

The Servlet Model

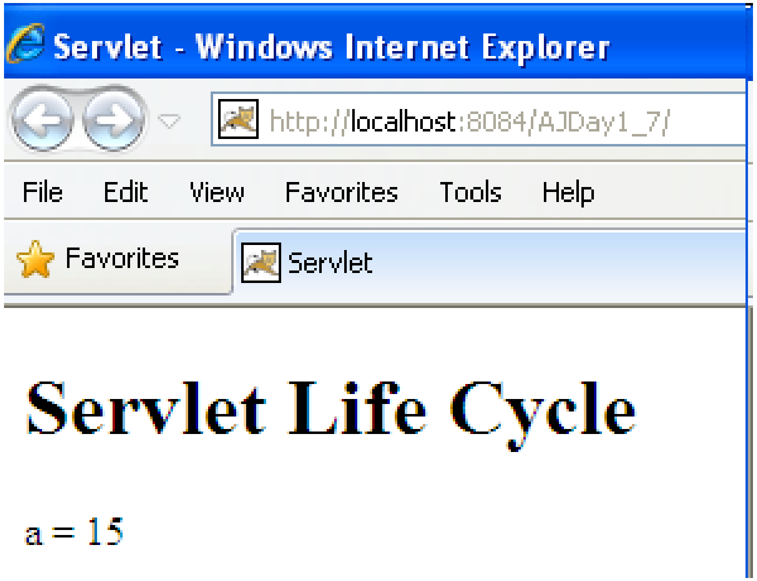
The Servlet Life Cycle – Example



```
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the
    /**...*/
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
        System.out.println("doGet is invoked");
    }

    /**...*/
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
        System.out.println("doPost is invoked");
    }
```

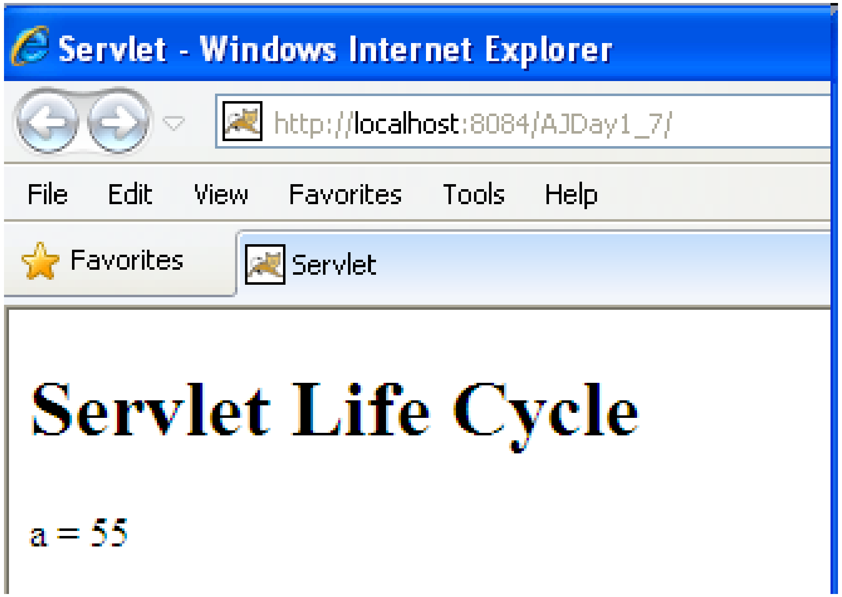
The Servlet Model

The Servlet Life Cycle – Example


Servlet - Windows Internet Explorer
http://localhost:8084/AJDay1_7/
File Edit View Favorites Tools Help
Favorites
Servlet

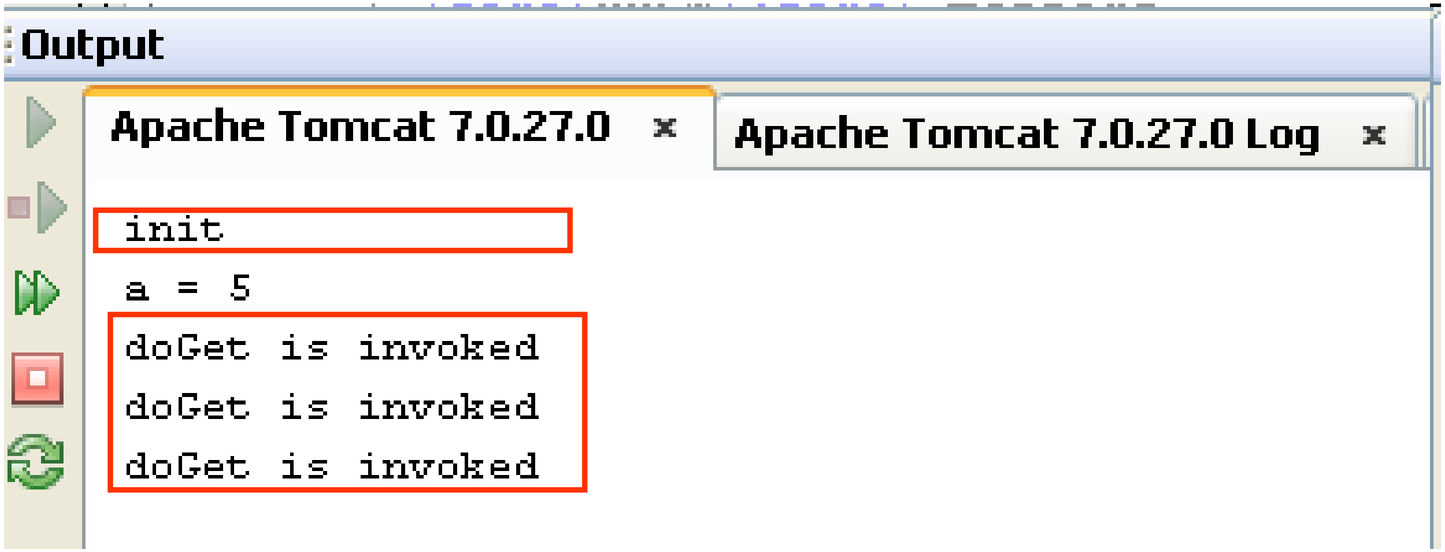
Servlet Life Cycle

a = 15


Servlet - Windows Internet Explorer
http://localhost:8084/AJDay1_7/
File Edit View Favorites Tools Help
Favorites
Servlet

Servlet Life Cycle

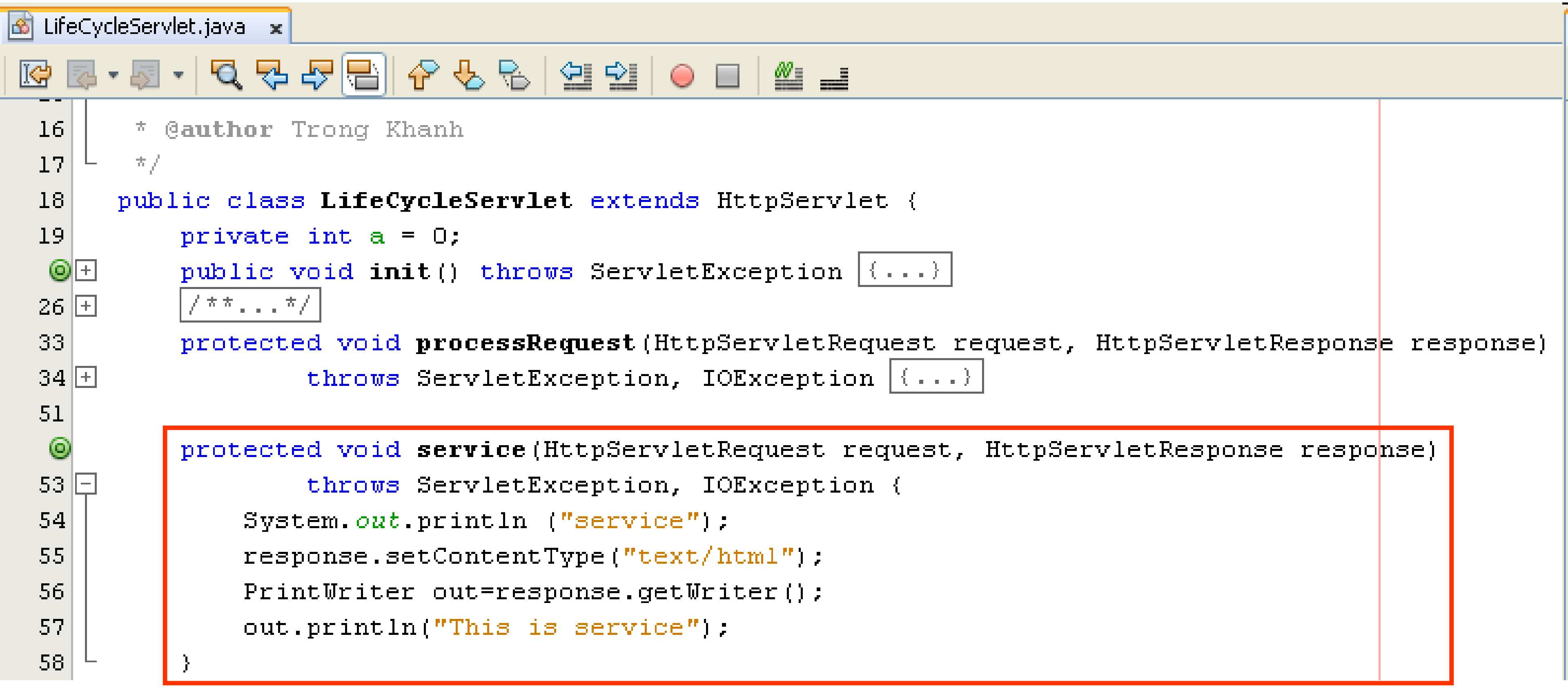
a = 55



```
init
a = 5
doGet is invoked
doGet is invoked
doGet is invoked
```

The Servlet Model

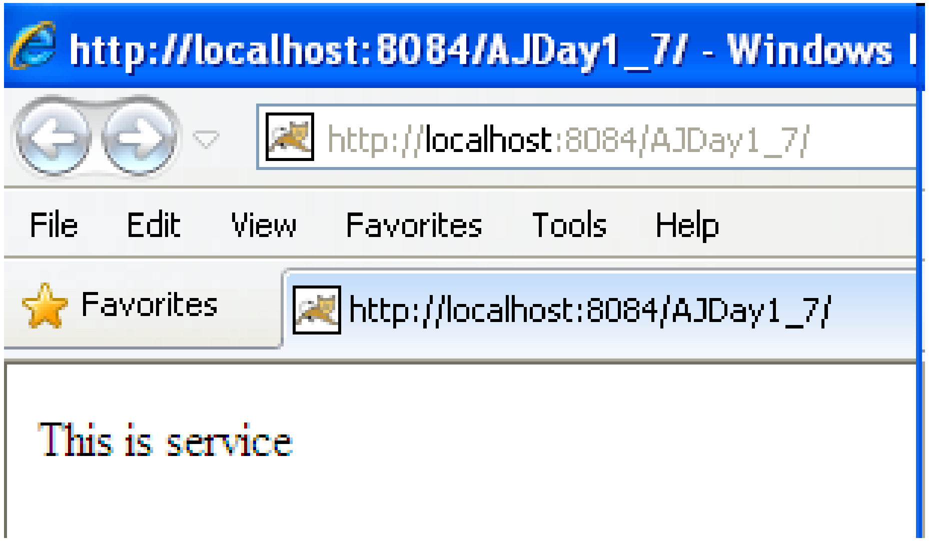
The Servlet Life Cycle – Example



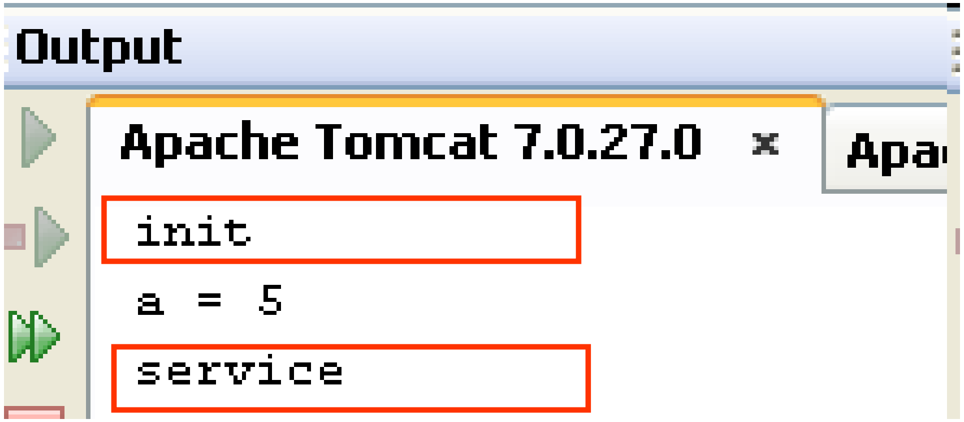
```
 * @author Trong Khanh
 */
public class LifeCycleServlet extends HttpServlet {
    private int a = 0;
    public void init() throws ServletException {...}
    /**...*/
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {...}

    protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println ("service");
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("This is service");
    }
```

The Servlet Model

The Servlet Life Cycle – Example


http://localhost:8084/AJDay1_7/ - Windows I
http://localhost:8084/AJDay1_7/
File
Edit
View
Favorites
Tools
Help
Favorites
http://localhost:8084/AJDay1_7/

This is service

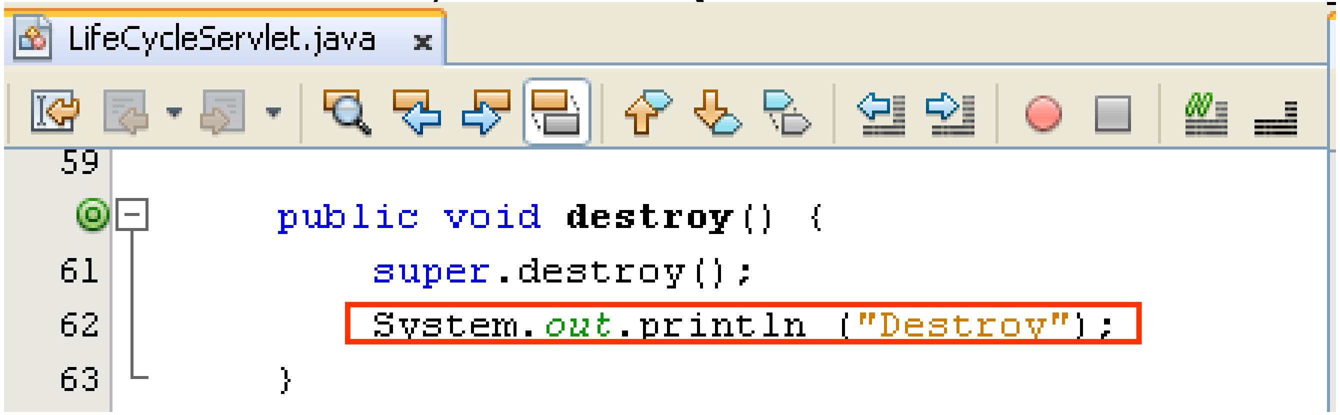


```
init
a = 5
service
```

The Servlet Model

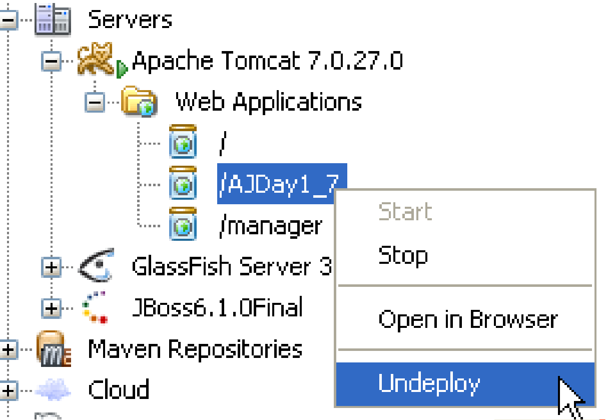
The Servlet Life Cycle – Example

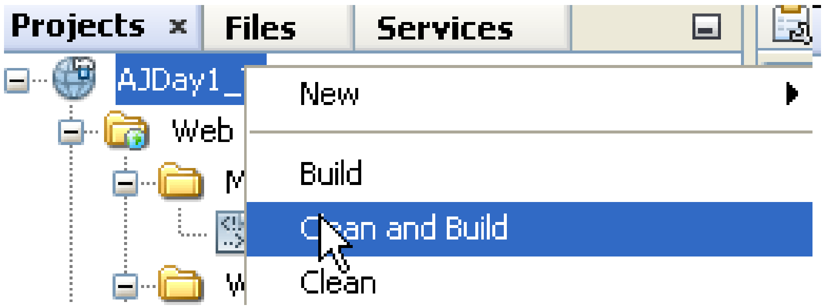
- **Addition the destroy method** (comment service method)

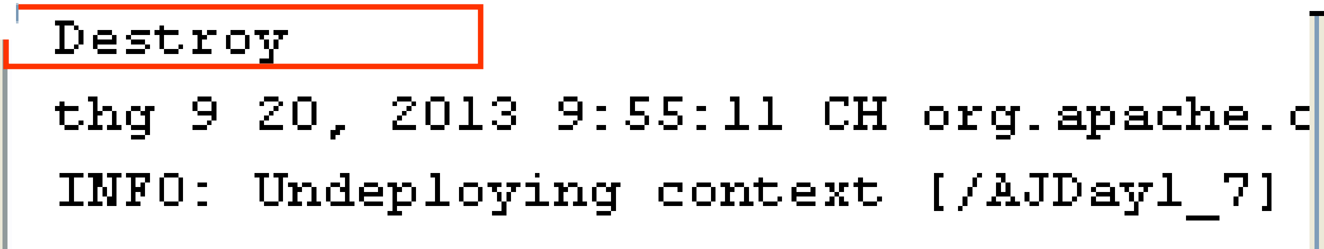


```
    public void destroy() {
        super.destroy();
        System.out.println ("Destroy");
    }
```

- **Execute project again, then undeploy or clean and Build the current project on Tomcat Server**



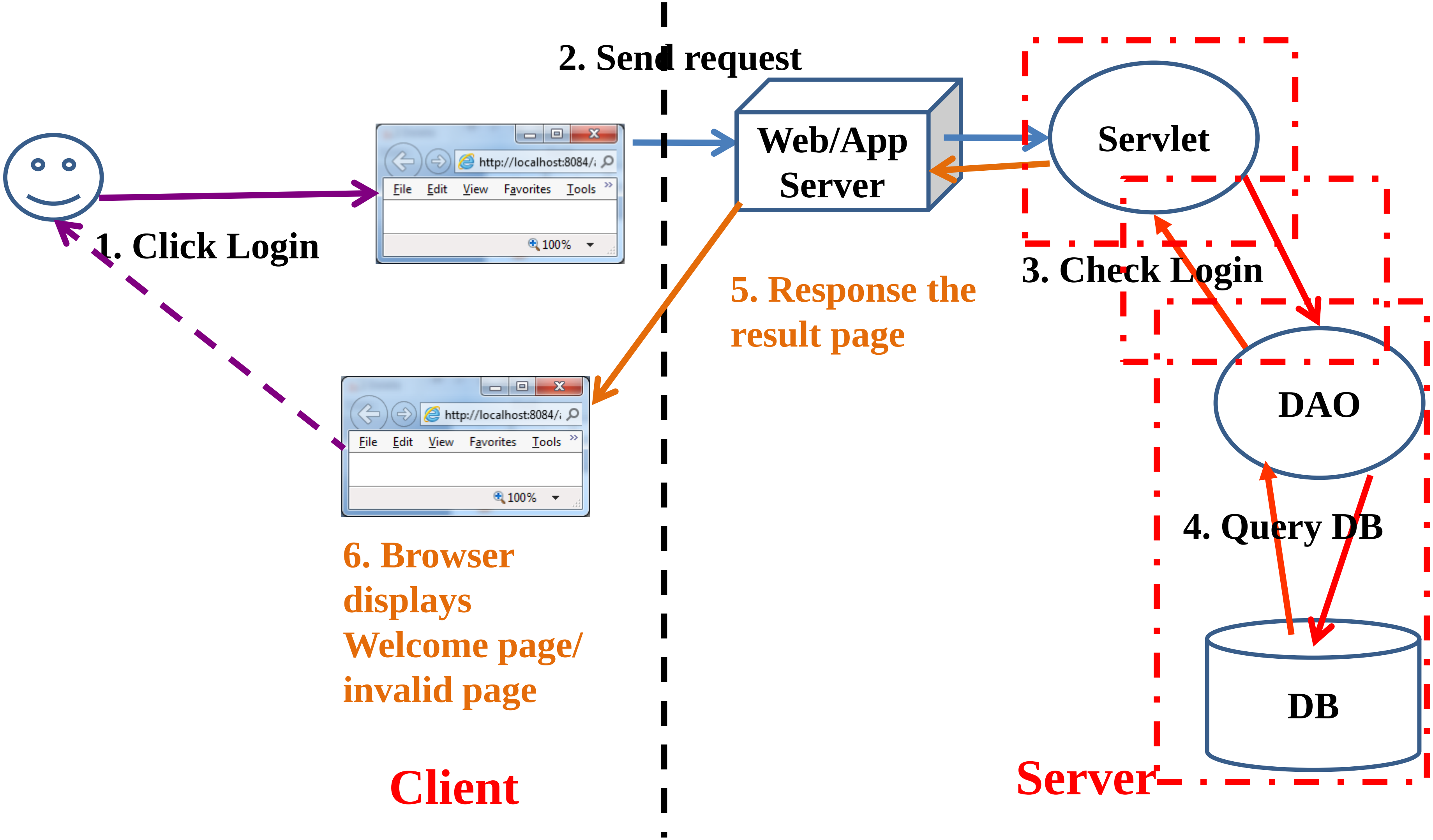




```
Destroy
thg 9 20, 2013 9:55:11 CH org.apache.c
INFO: Undeploying context [/AJDay1_7]
```

Build The Simple Web

Interactive Server Model


http://localhost:8084/
File Edit View Favorites Tools
100%
http://localhost:8084/
File Edit View Favorites Tools
100%

1. Click Login

2. Send request

Web/App Server

Servlet

3. Check Login

DAO

4. Query DB

DB

5. Response the result page

6. Browser displays Welcome page/ invalid page

Client

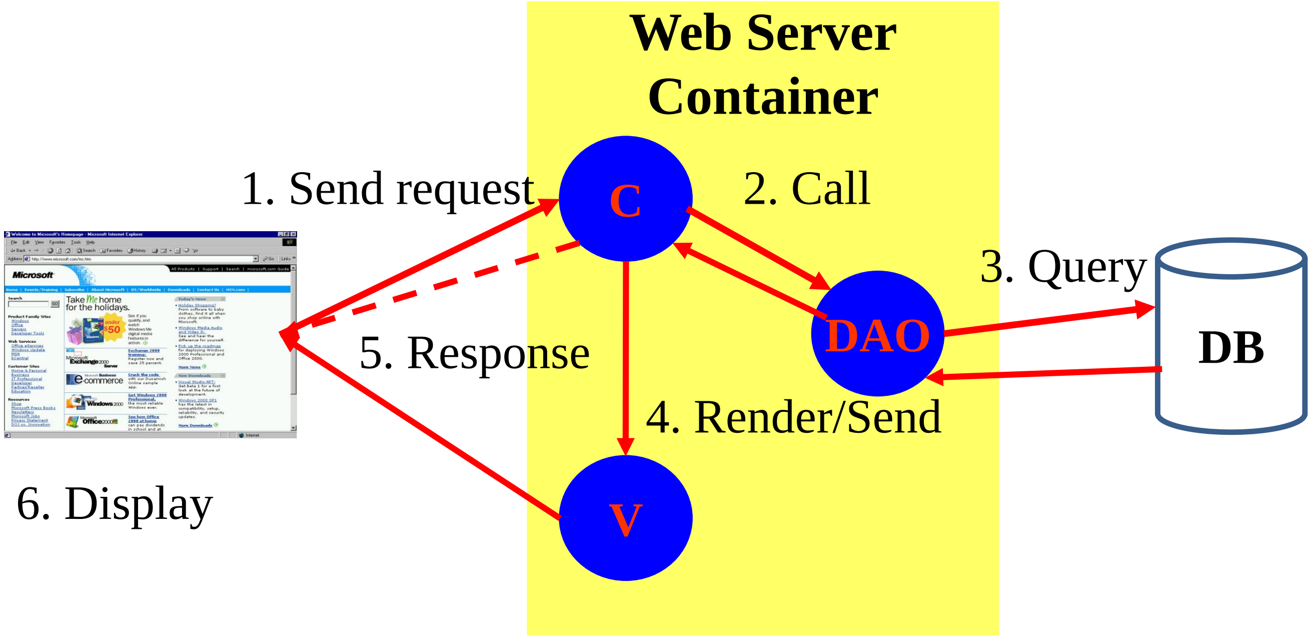
Server

Summary

- **How to build the simple web site using html and servlet?**
  - Http Protocol and Methods
  - What is Servlet?
  - Parameters vs. Variables
  - Servlet Life Cycle
  - Break down structure component in building web application

Q&A

Summary



Web Server Container

1. Send request

2. Call

3. Query

4. Render/Send

5. Response

6. Display

Q&A

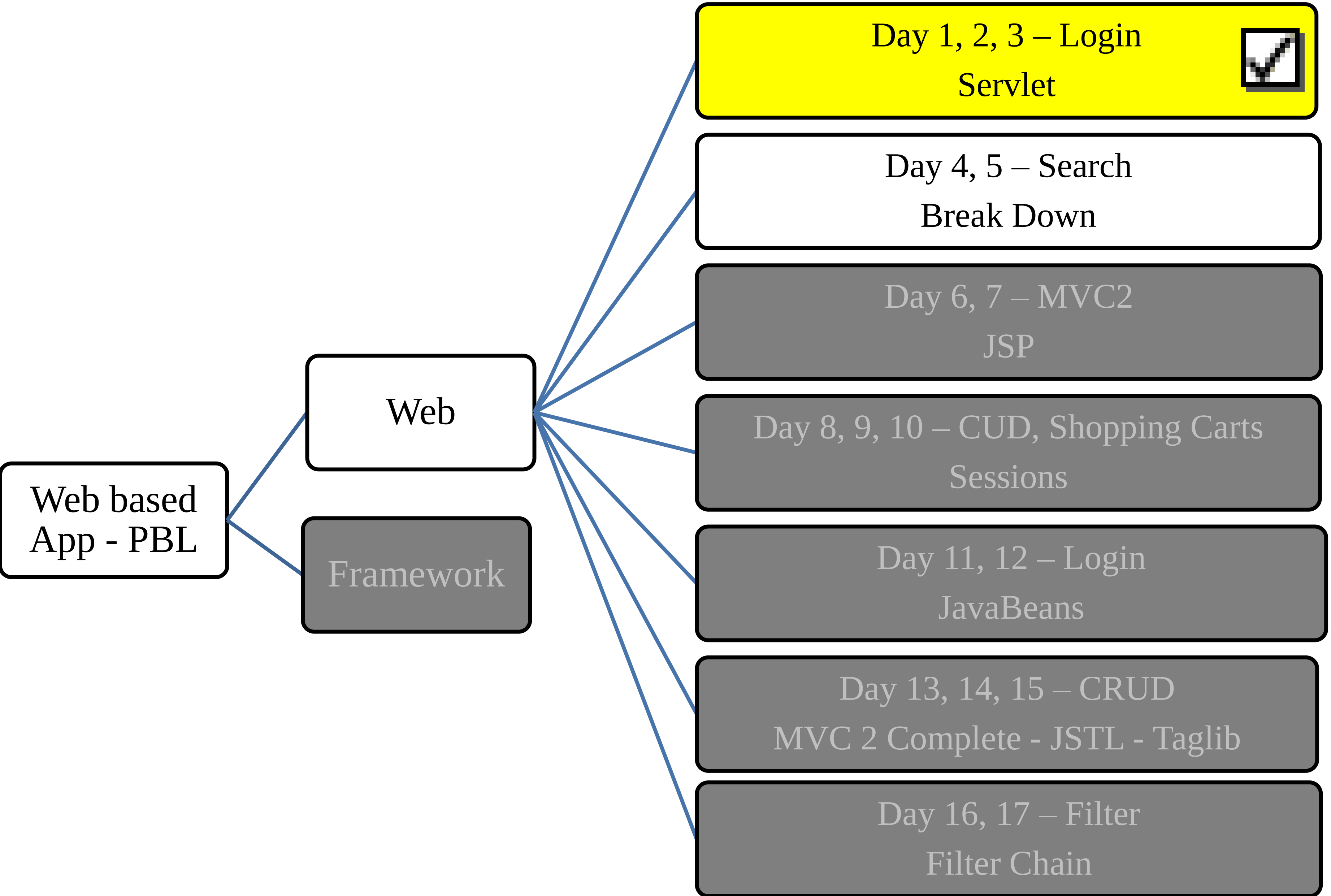
Exercises

- Do it again all of demos
- Using servlet to write the programs as the following requirement
  - Present the Login form (naming LoginServlet) with title Login, header h1 – Login, 02 textbox with naming txtUser and txtPass, and the Login button
    - Rewrite above Login application combining with DB
  - Writing the ColorServlet that presents “Welcome to Servlet course” with yellow in background and red in foreground
  - Writing the ProductServlet includes a form with a combo box containing Servlet & JSP, Struts & JSF, EJB, XMJ, Java Web Services, and the button with value Add to Cart

Next Lecture

- **How to deploy the Web Application to Web Server?**
  - Web applications Structure
  - Request Parameters vs. Context Parameters vs. Config/Servlet Parameters
  - Application Segments vs. Scope
- **How to transfer from resources to others with/without data/objects?**
  - Attributes vs. Parameters vs. Variables
  - Redirect vs. RequestDispatcher
  - RequestDispatcher vs. Filter

Next Lecture



Web based App - PBL

Web

Framework

Day 1, 2, 3 – Login
Servlet ☑

Day 4, 5 – Search
Break Down

Day 6, 7 – MVC2
JSP

Day 8, 9, 10 – CUD, Shopping Carts
Sessions

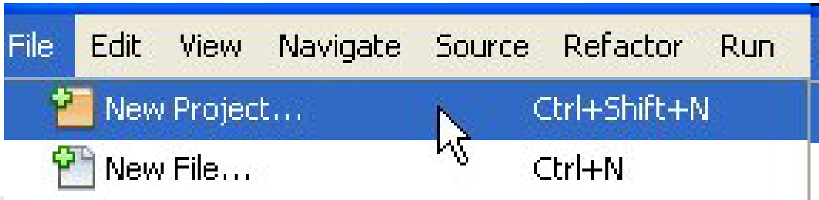
Day 11, 12 – Login
JavaBeans

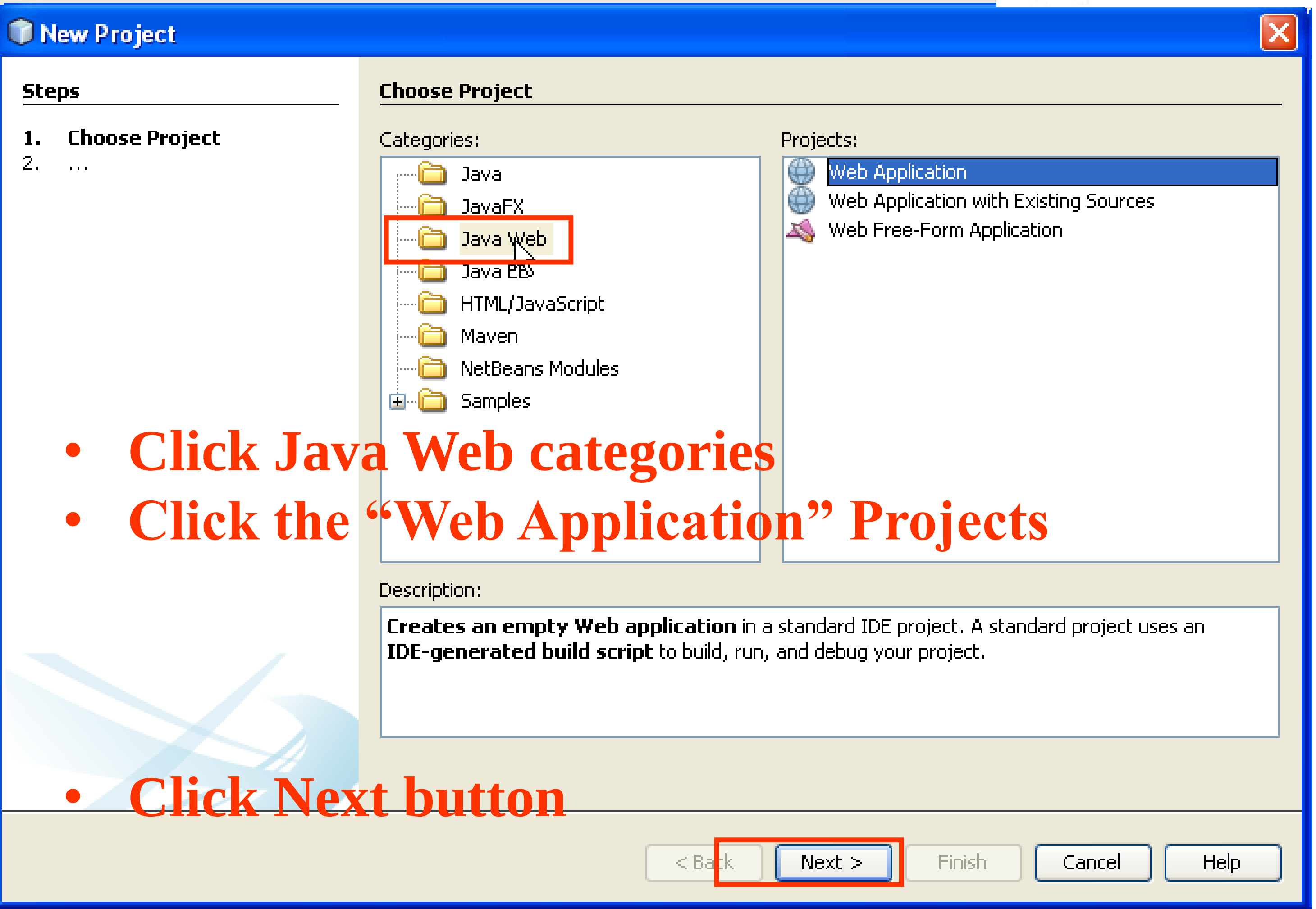
Day 13, 14, 15 – CRUD
MVC 2 Complete - JSTL - Taglib

Day 16, 17 – Filter
Filter Chain

Appendix – Build The Simple Web

How to Create Web Application Project



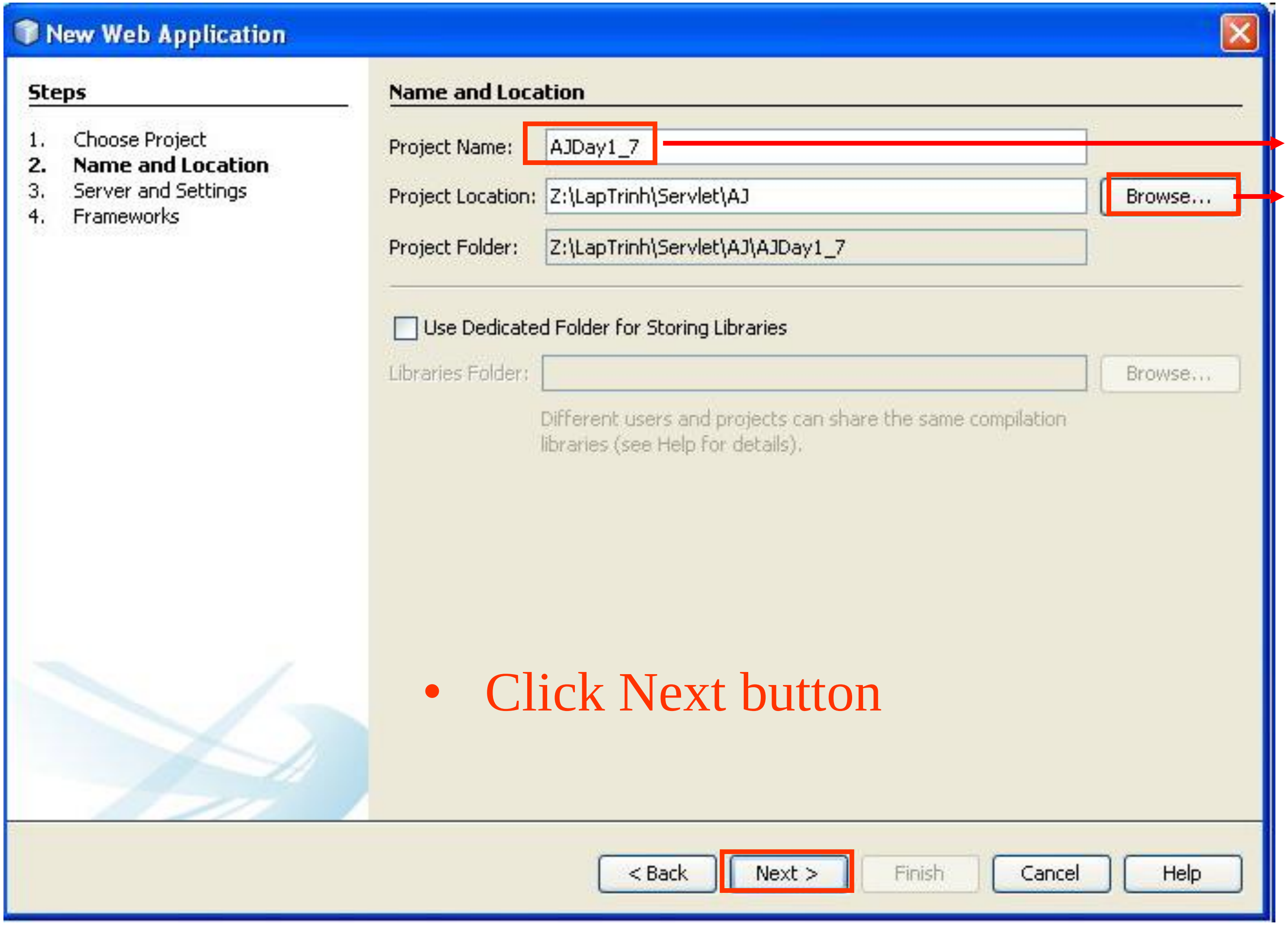


Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

- **Click Java Web categories**
- **Click the "Web Application" Projects**
- **Click Next button**

Appendix – Build The Simple Web

How to Create Web Application Project



New Web Application

Steps

1. Choose Project
2. **Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name: AJDay1_7

Project Location: Z:\LapTrinh\Servlet\AJ

Project Folder: Z:\LapTrinh\Servlet\AJ\AJDay1_7

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

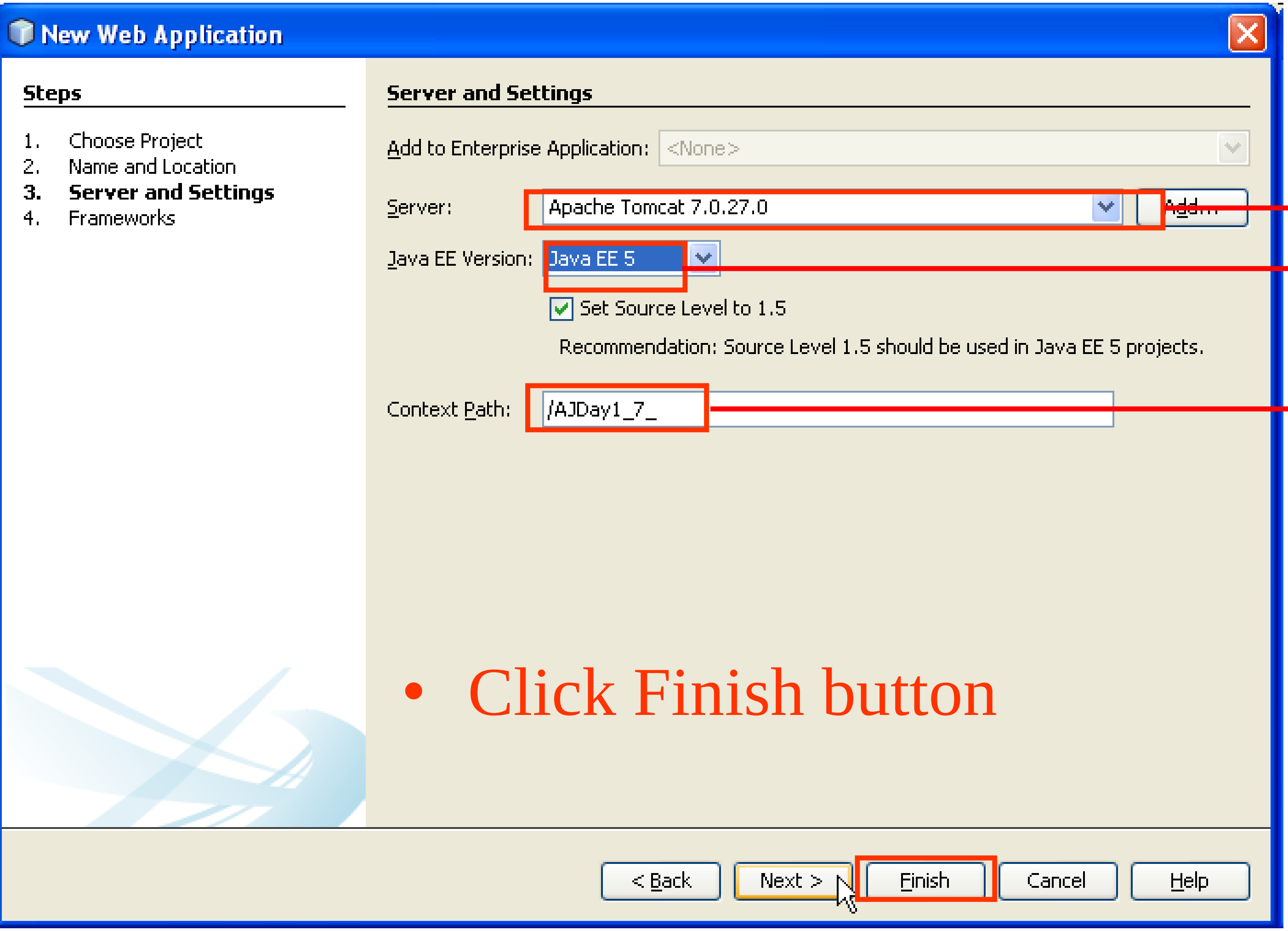
Fill your project name

Browser your location where store the project

- Click Next button

Appendix – Build The Simple Web

How to Create Web Application Project



New Web Application

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

Add to Enterprise Application: <None>

Server: Apache Tomcat 7.0.27.0

Java EE Version: Java EE 5

☑ Set Source Level to 1.5

Recommendation: Source Level 1.5 should be used in Java EE 5 projects.

Context Path: /AJDay1_7_

Choose deployed server

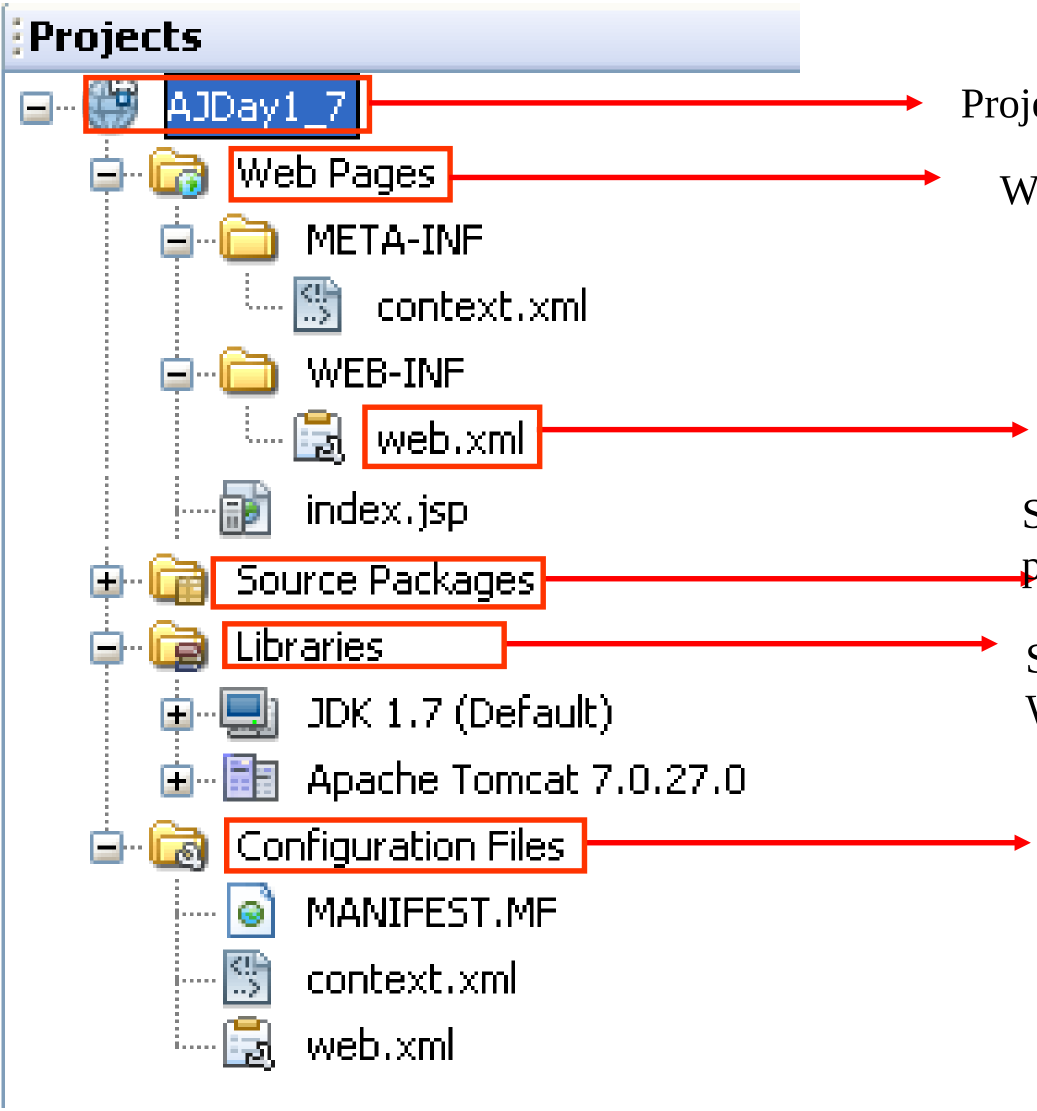
Choose Java EE 5/ J2EE 1.4

Modify the context path (if necessary). Defaults, it is named same as Project Name

- Click Finish button

Appendix – Build The Simple Web

How to Create Web Application Project



Projects

- AJDay1_7
  - Web Pages
    - META-INF
      - context.xml
    - WEB-INF
      - web.xml
    - index.jsp
  - Source Packages
  - Libraries
    - JDK 1.7 (Default)
    - Apache Tomcat 7.0.27.0
  - Configuration Files
    - MANIFEST.MF
    - context.xml
    - web.xml

Project name

Web Directory

Web deployment descriptor

Source code directory, containing **java class**. When project is built, package in **classes directory**

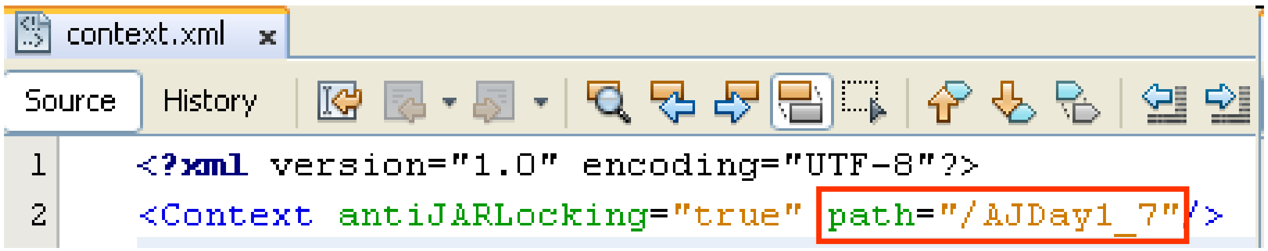
Support library directory, containing **jar file**. When project is built, package in **lib directory**

Configuration directory related define for Web App

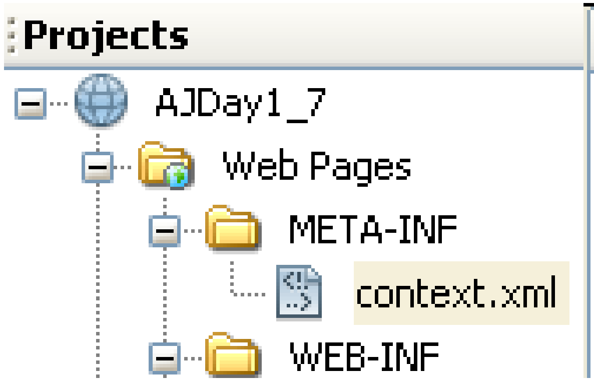
Appendix – Web Applications

Add the META-INF/context.xml to project

- *optional – if it does not exist*
  - **Right click** the **Web Pages**, choose **New**, **then** choose **Other**
  - In New **File Dialog**, **choose Other**, then choose **Folder, click Next**
  - In New **Folder Dialog**, type the **META-INF** into Folder Name
  - Click **Finish**
  - **Right click the META-INF**, choose **New**, **then** choose **Other**
  - In New File Dialog, **choose XML**, then choose **XML Document**, click **Next**
  - In New XML Document Dialog, **type context** into **File Name, click Next, then click Finish**
  - Type the **content of content.xml file** as (**Notes: must type "/" in front of context**)

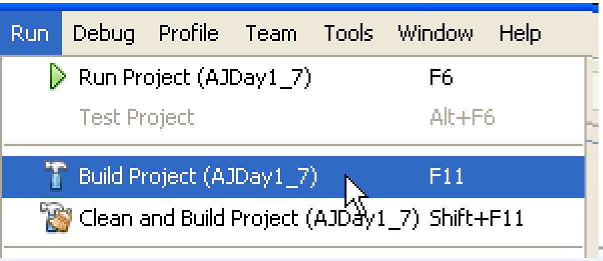


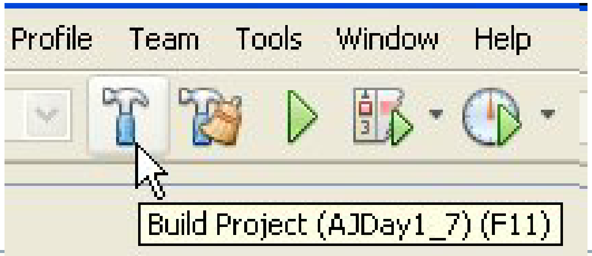
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/AJDay1_7"/>
```

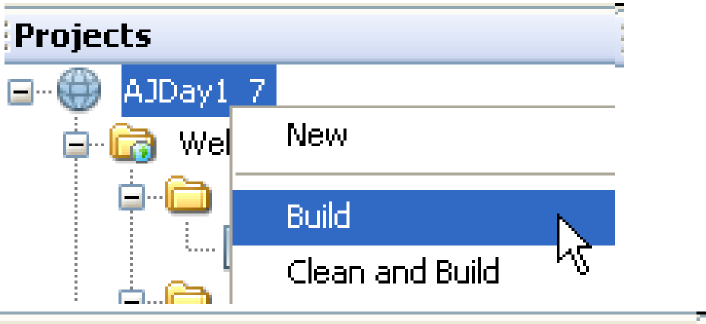


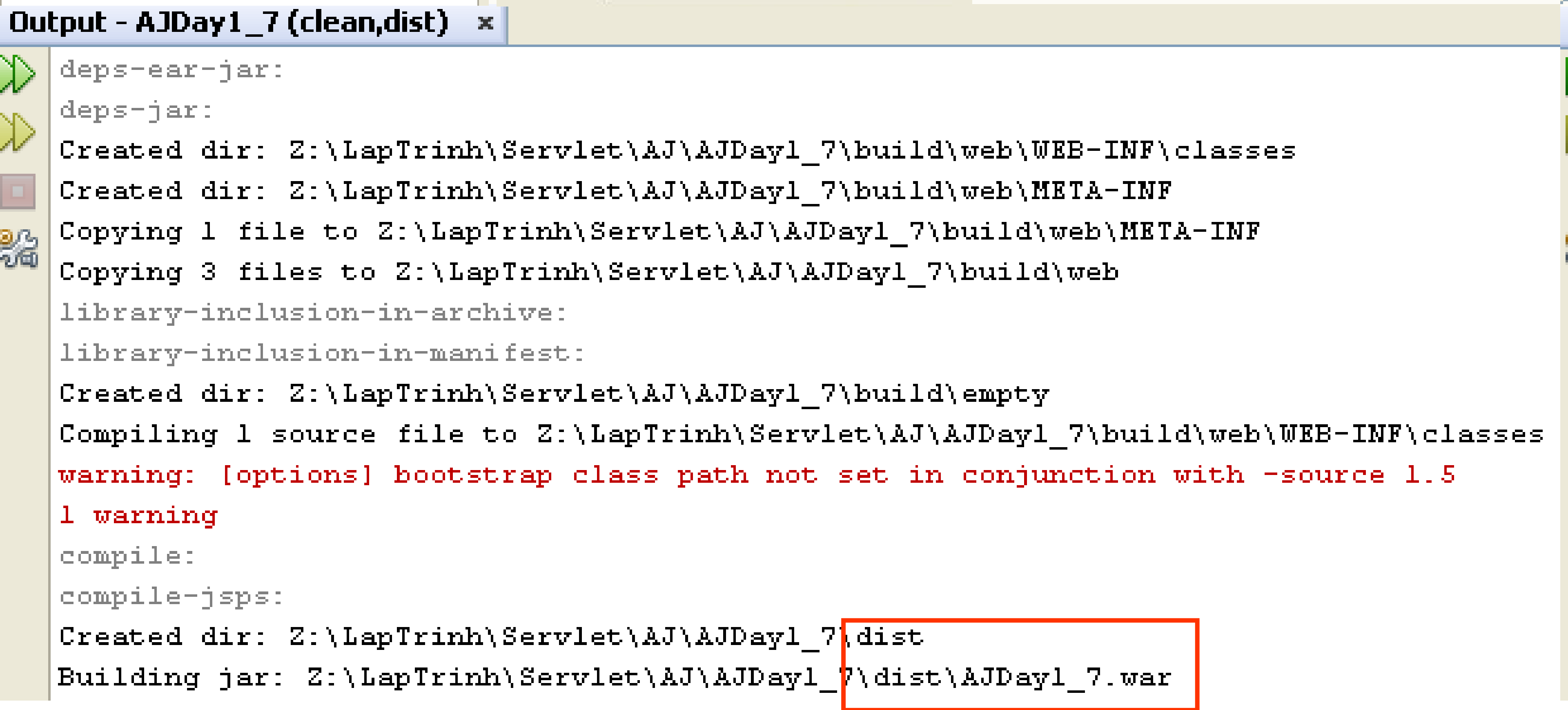
Appendix

Build Application


Run
Debug
Profile
Team
Tools
Window
Help
Run Project (AJDay1_7) F6
Test Project Alt+F6
Build Project (AJDay1_7) F11
Clean and Build Project (AJDay1_7) Shift+F11



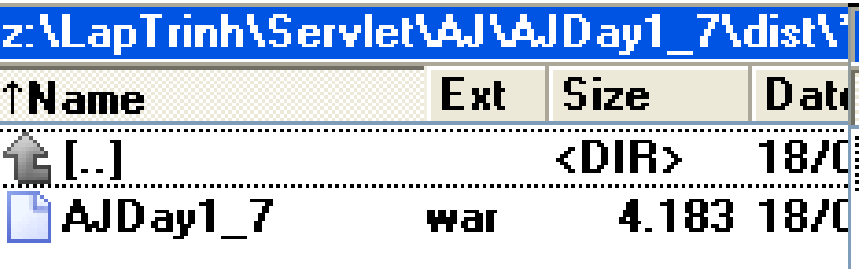




```
deps-ear-jar:
deps-jar:
Created dir: Z:\LapTrinh\Servlet\AJ\AJDay1_7\build\web\WEB-INF\classes
Created dir: Z:\LapTrinh\Servlet\AJ\AJDay1_7\build\web\META-INF
Copying 1 file to Z:\LapTrinh\Servlet\AJ\AJDay1_7\build\web\META-INF
Copying 3 files to Z:\LapTrinh\Servlet\AJ\AJDay1_7\build\web
library-inclusion-in-archive:
library-inclusion-in-manifest:
Created dir: Z:\LapTrinh\Servlet\AJ\AJDay1_7\build\empty
Compiling 1 source file to Z:\LapTrinh\Servlet\AJ\AJDay1_7\build\web\WEB-INF\classes
warning: [options] bootstrap class path not set in conjunction with -source 1.5
1 warning
compile:
compile-jsps:
Created dir: Z:\LapTrinh\Servlet\AJ\AJDay1_7\dist
Building jar: Z:\LapTrinh\Servlet\AJ\AJDay1_7\dist\AJDay1_7.war
```

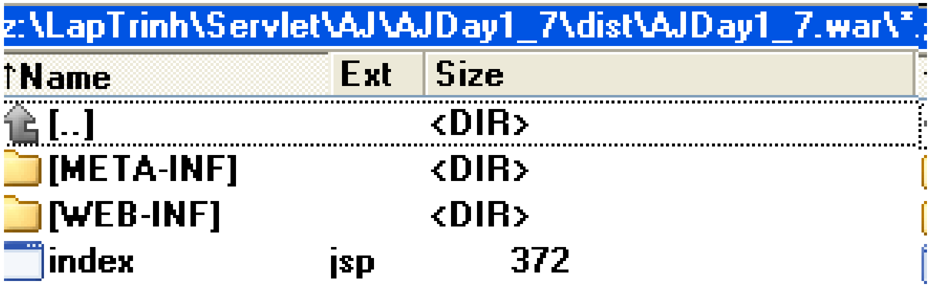
Appendix

Build Application



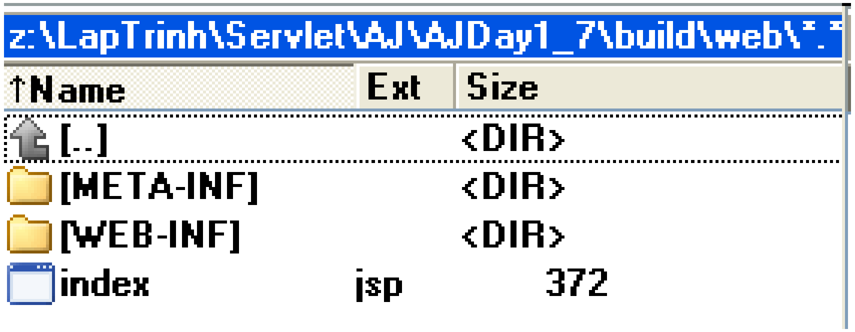
z:\LapTrinh\Servlet\AJ\AJDay1_7\dist\*

↑Name	Ext	Size	Dat
[..]		<DIR>	18/0
AJDay1_7	war	4.183	18/0



z:\LapTrinh\Servlet\AJ\AJDay1_7\dist\AJDay1_7.war*.

↑Name	Ext	Size
[..]		<DIR>
[META-INF]		<DIR>
[WEB-INF]		<DIR>
index	jsp	372



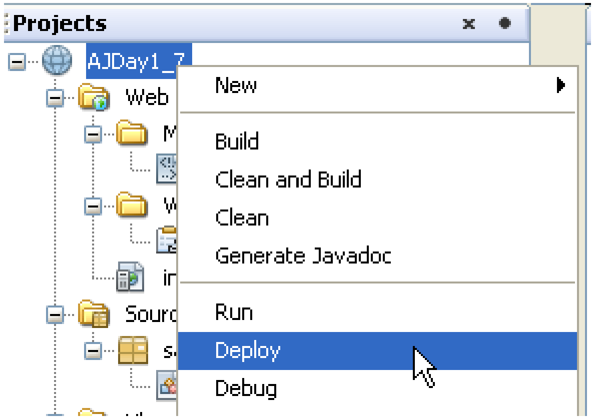
z:\LapTrinh\Servlet\AJ\AJDay1_7\build\web\*.*

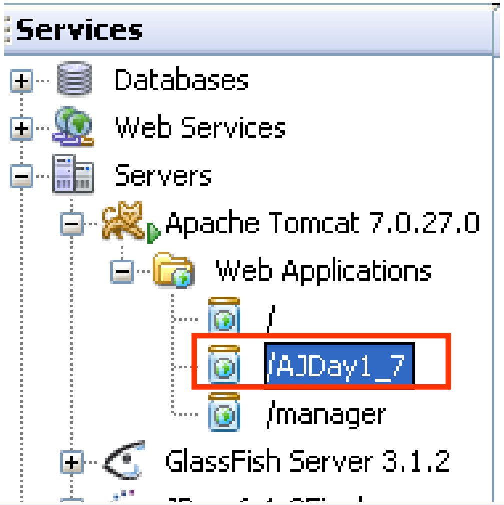
↑Name	Ext	Size
[..]		<DIR>
[META-INF]		<DIR>
[WEB-INF]		<DIR>
index	jsp	372

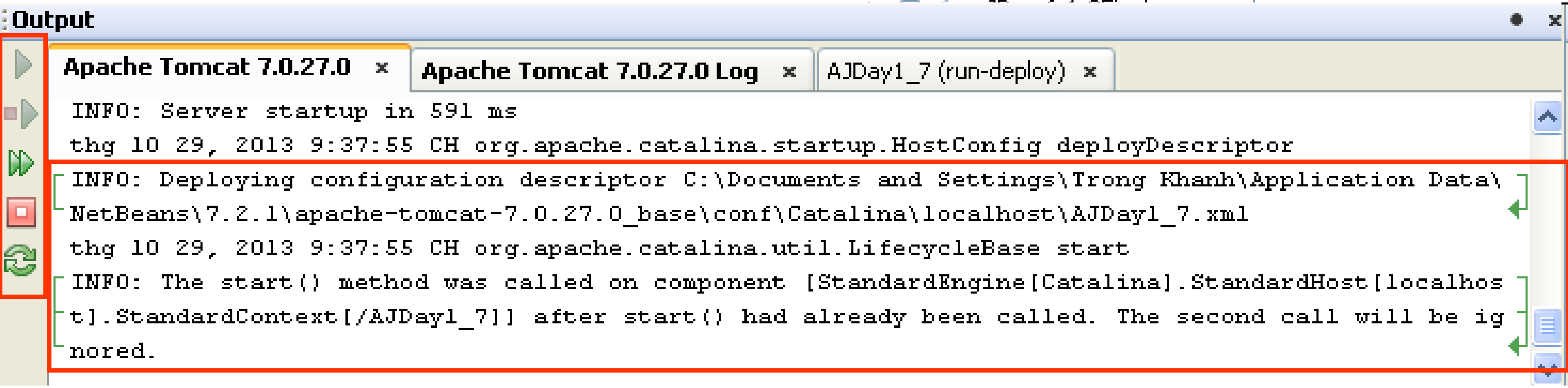
- **Package War file** with **command prompt**
  - **jar –cvf** fileName.war directoryOrFile
  - **Ex**: jar –cvf AJDay1_7.war *.jsp WEB-INF/*

Appendix

Deploy Application



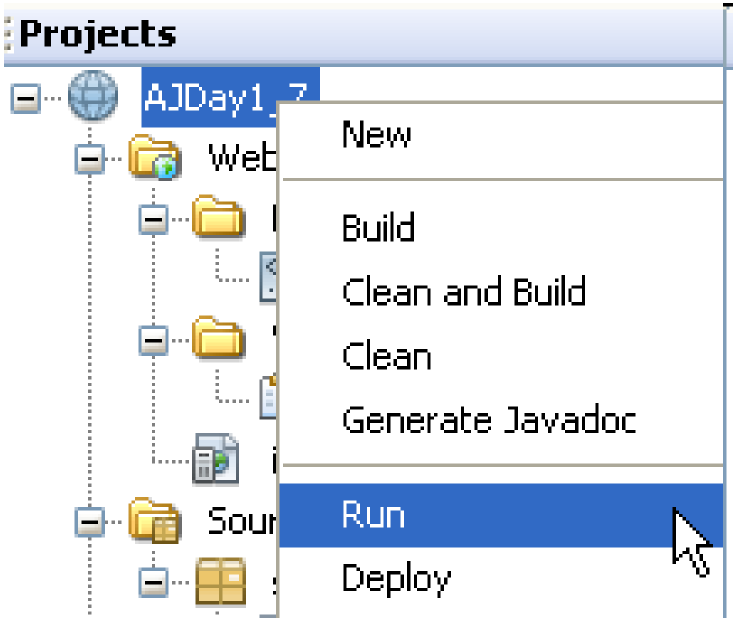


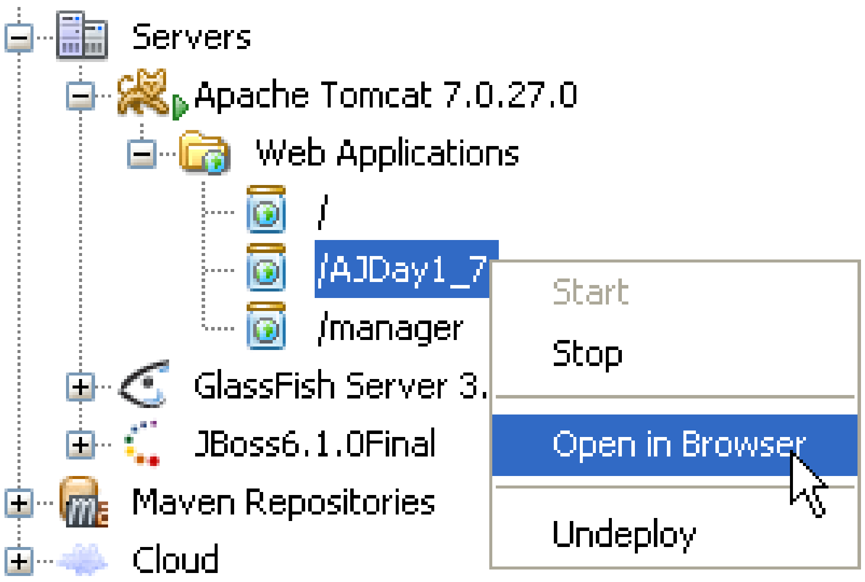


```
INFO: Server startup in 591 ms
thg 10 29, 2013 9:37:55 CH org.apache.catalina.startup.HostConfig deployDescriptor
INFO: Deploying configuration descriptor C:\Documents and Settings\Trong Khanh\Application Data\
NetBeans\7.2.1\apache-tomcat-7.0.27.0_base\conf\Catalina\localhost\AJDay1_7.xml
thg 10 29, 2013 9:37:55 CH org.apache.catalina.util.LifecycleBase start
INFO: The start() method was called on component [StandardEngine[Catalina].StandardHost[localhos
t].StandardContext[/AJDay1_7]] after start() had already been called. The second call will be ig
nored.
```

Appendix

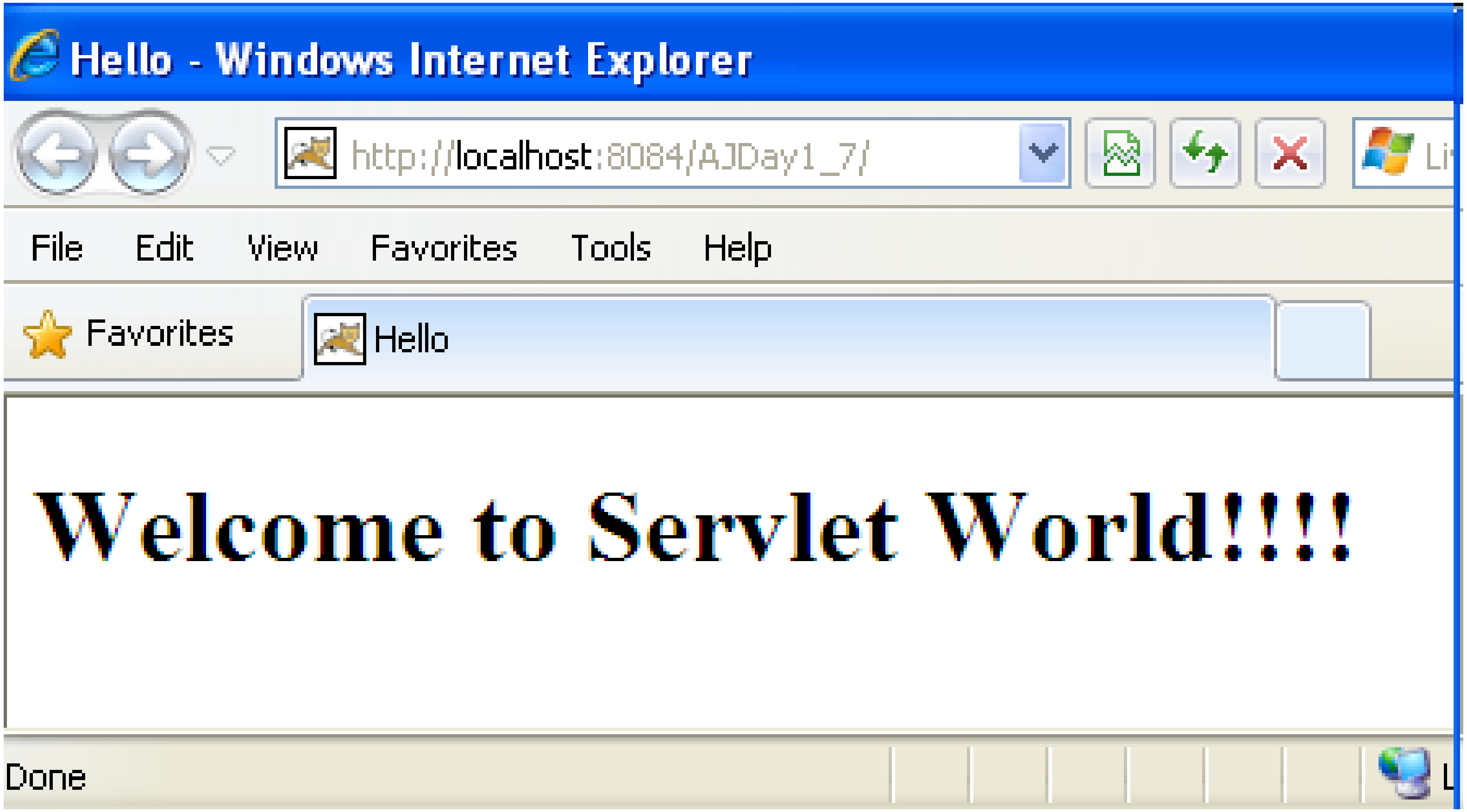
Run Application





Appendix

Run Application

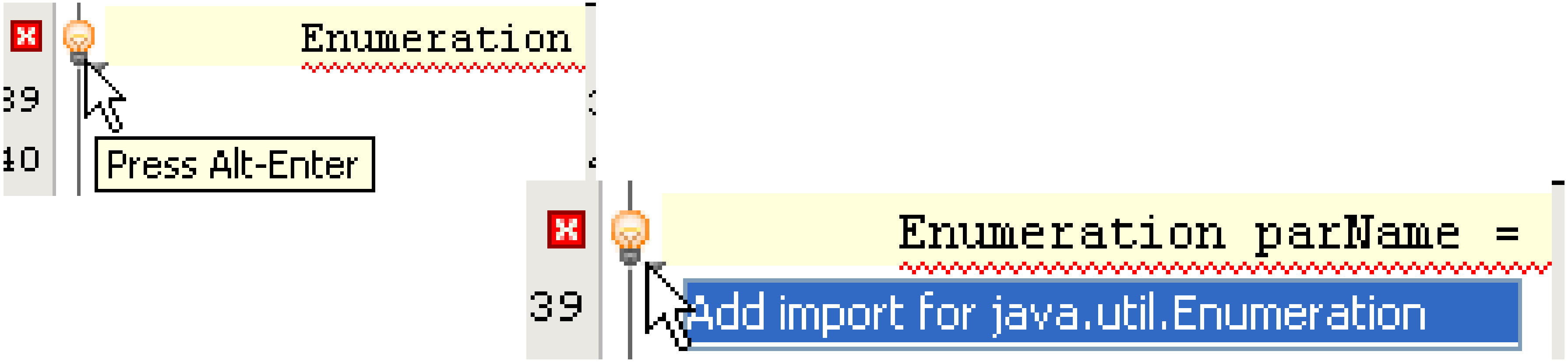


Hello - Windows Internet Explorer
http://localhost:8084/AJDay1_7/
File Edit View Favorites Tools Help
Favorites
Hello
Done

Welcome to Servlet World!!!!

Appendix

Additional

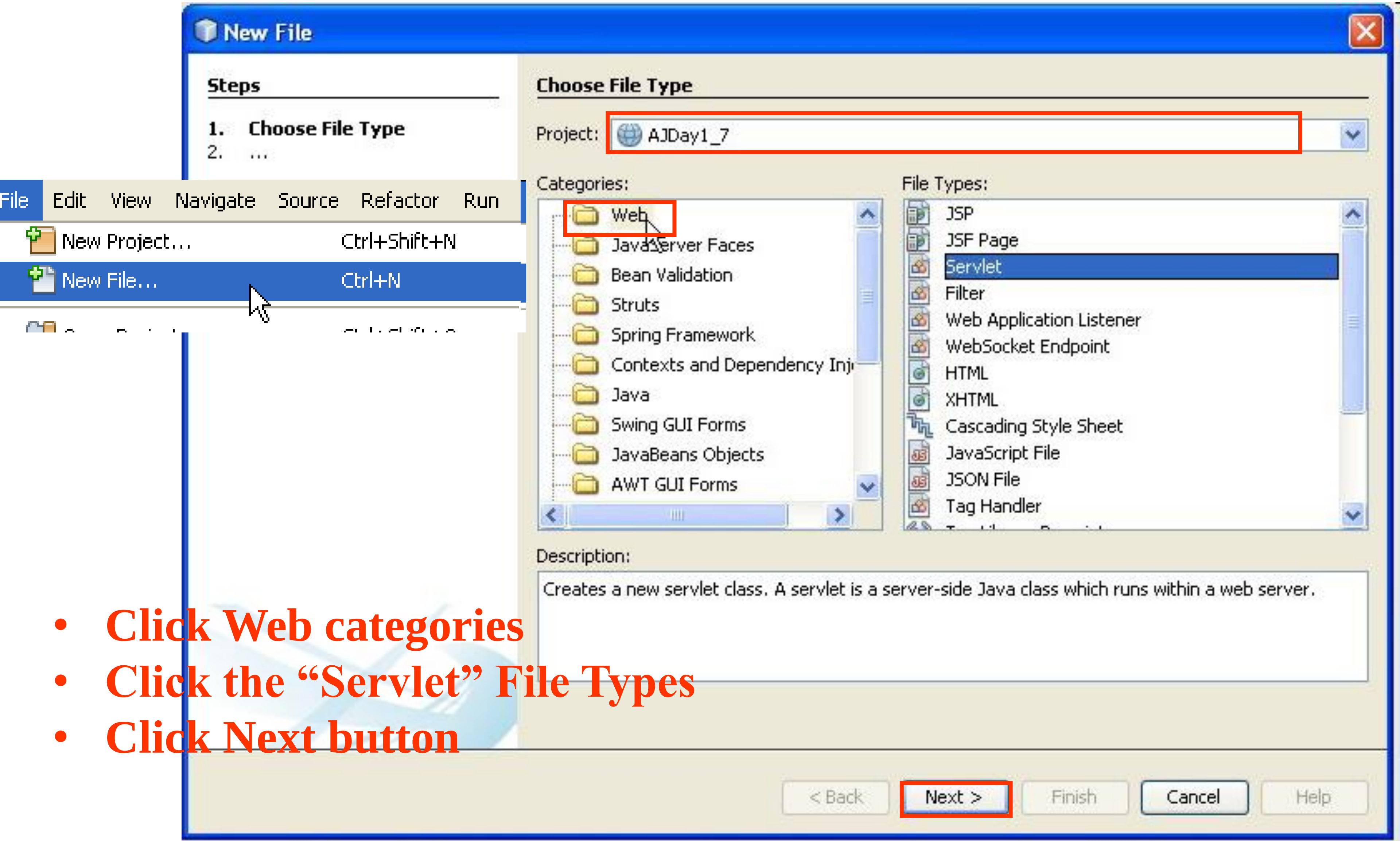


```
Enumeration
```

```
Enumeration parName =
```

- **Caches of server**
  - **WinXP**: C:\Documents and Settings\LoggedUser\Application Data\NetBeans\version\apache-tomcat-tomcatVersion_base\work\Catalina \localhost\
  - **Vista or Win7, 8, 10**: C:\Users\LoggedUser\AppData\Roaming\NetBeans\version\apache-tomcat-tomcatVersion_base\work\Catalina\localhost\
  - Above location should be **gone and cleared** when the application **cannot** be **undeployed** or the web servers **occur the errors**

Appendix
Create a Servlet



New File

Steps

1. Choose File Type
2. ...

File Edit View Navigate Source Refactor Run

New Project... Ctrl+Shift+N

New File... Ctrl+N

Choose File Type

Project: AJDay1_7

Categories:

- Web
- JavaServer Faces
- Bean Validation
- Struts
- Spring Framework
- Contexts and Dependency Inj
- Java
- Swing GUI Forms
- JavaBeans Objects
- AWT GUI Forms

File Types:

- JSP
- JSF Page
- Servlet
- Filter
- Web Application Listener
- WebSocket Endpoint
- HTML
- XHTML
- Cascading Style Sheet
- JavaScript File
- JSON File
- Tag Handler

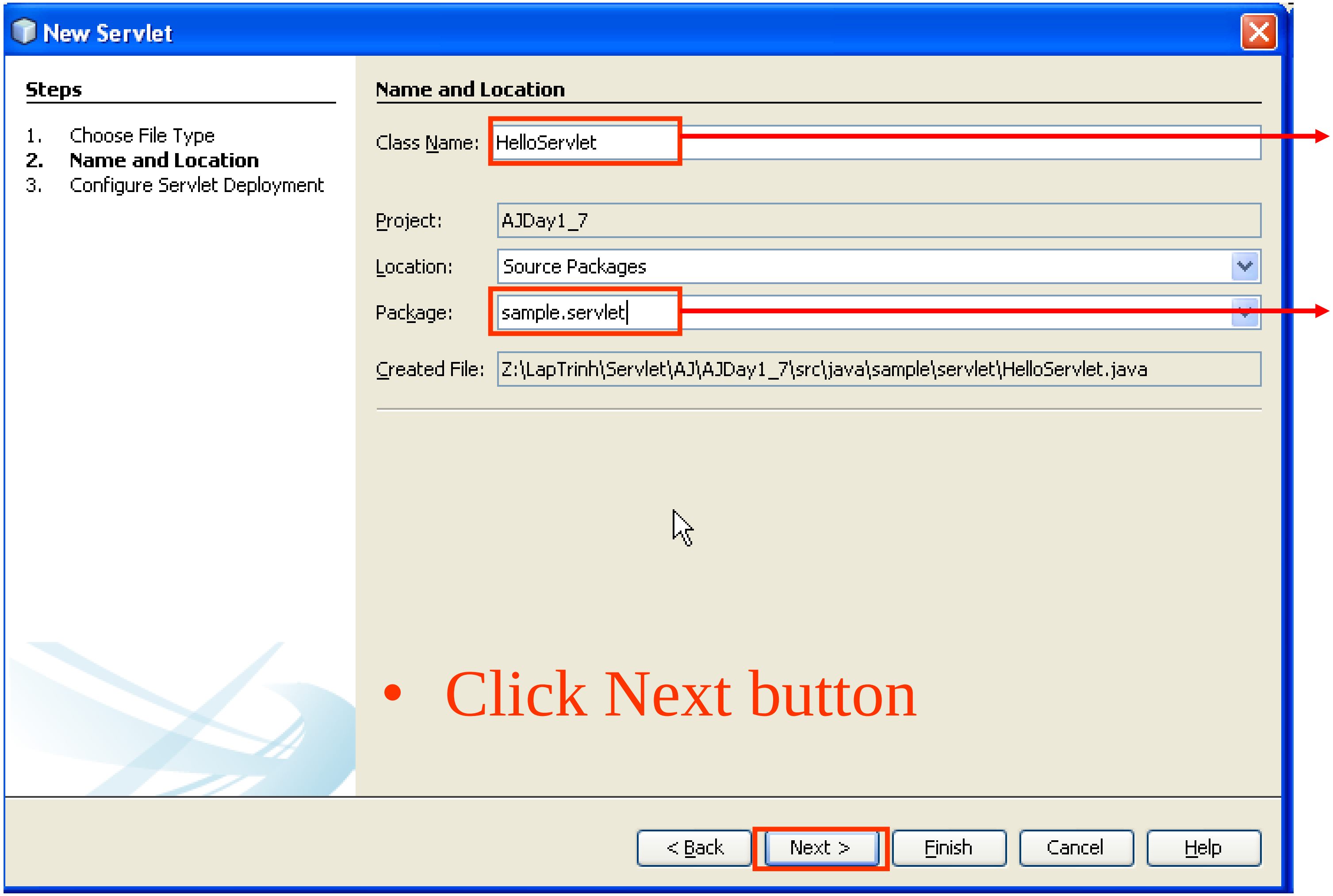
Description:

Creates a new servlet class. A servlet is a server-side Java class which runs within a web server.

< Back | Next > | Finish | Cancel | Help

- **Click Web categories**
- **Click the "Servlet" File Types**
- **Click Next button**

Appendix
Create a Servlet



Steps

1. Choose File Type
2. **Name and Location**
3. Configure Servlet Deployment

Name and Location

Class Name: HelloServlet

Project: AJDay1_7

Location: Source Packages

Package: sample.servlet

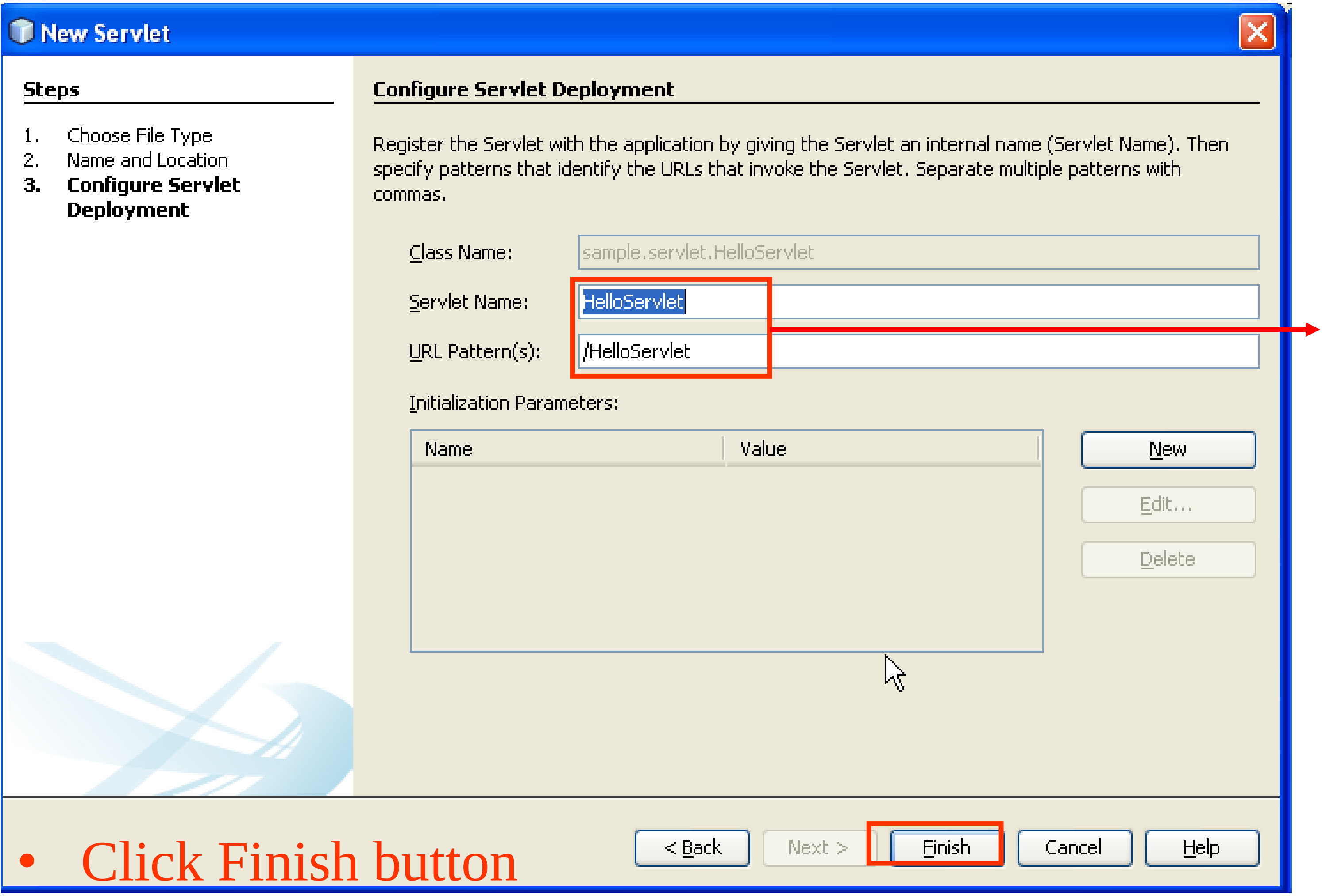
Created File: Z:\LapTrinh\Servlet\AJ\AJDay1_7\src\java\sample\servlet\HelloServlet.java

Fill your servlet name

Fill or choose package name

- Click Next button

Appendix
Create a Servlet



New Servlet

Steps

1. Choose File Type
2. Name and Location
3. **Configure Servlet Deployment**

Configure Servlet Deployment

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

Class Name: sample.servlet.HelloServlet

Servlet Name: HelloServlet

URL Pattern(s): /HelloServlet

Initialization Parameters:

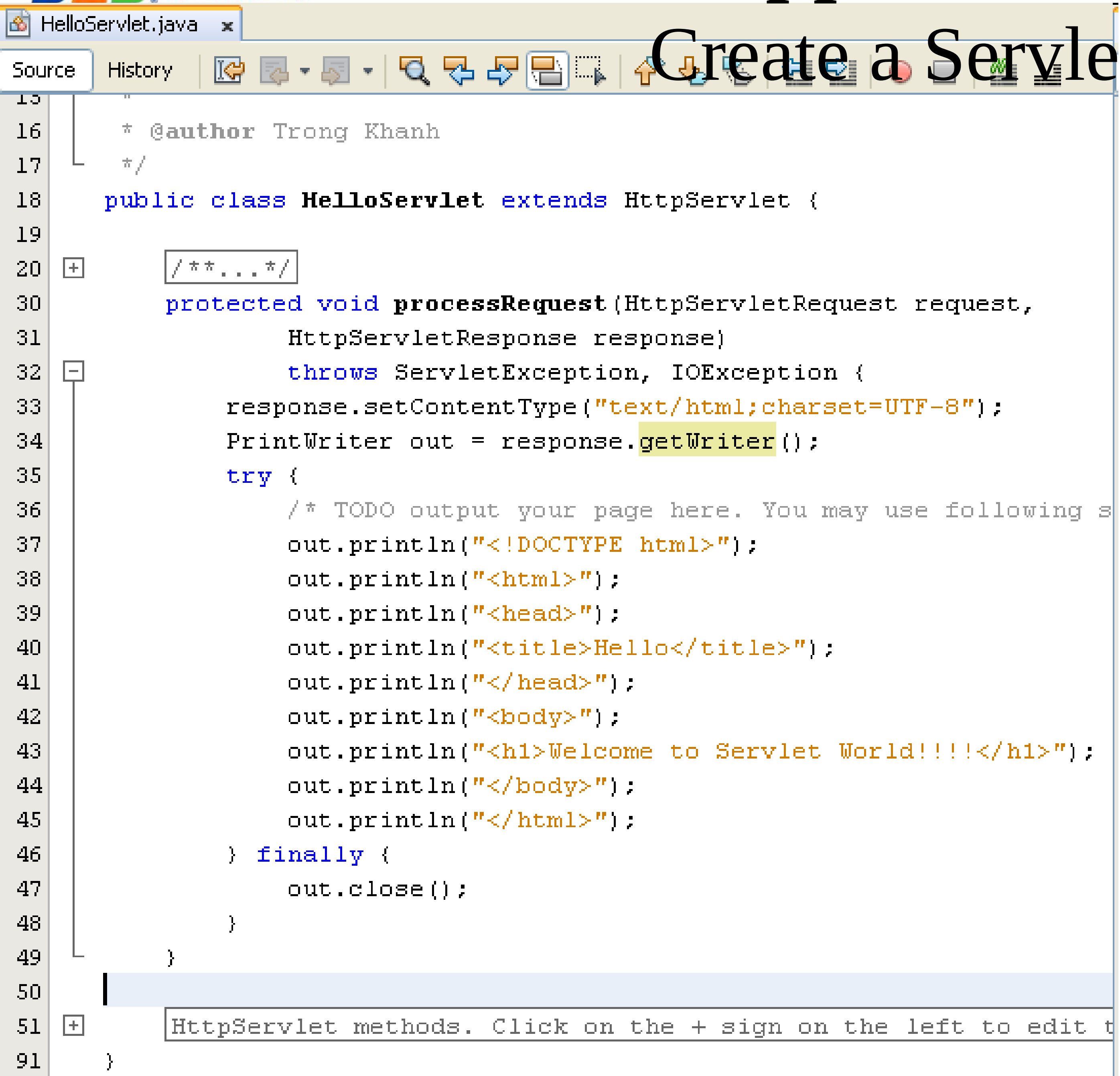
Name	Value

Modify the Servlet Name or URL Pattern if necessary) to configure the servlet information to web.xml

- Click Finish button
- The servlet class (ex: HelloServlet.java) is added to source packages (with package name if it’s exist) and it’s information is added to xml

Appendix

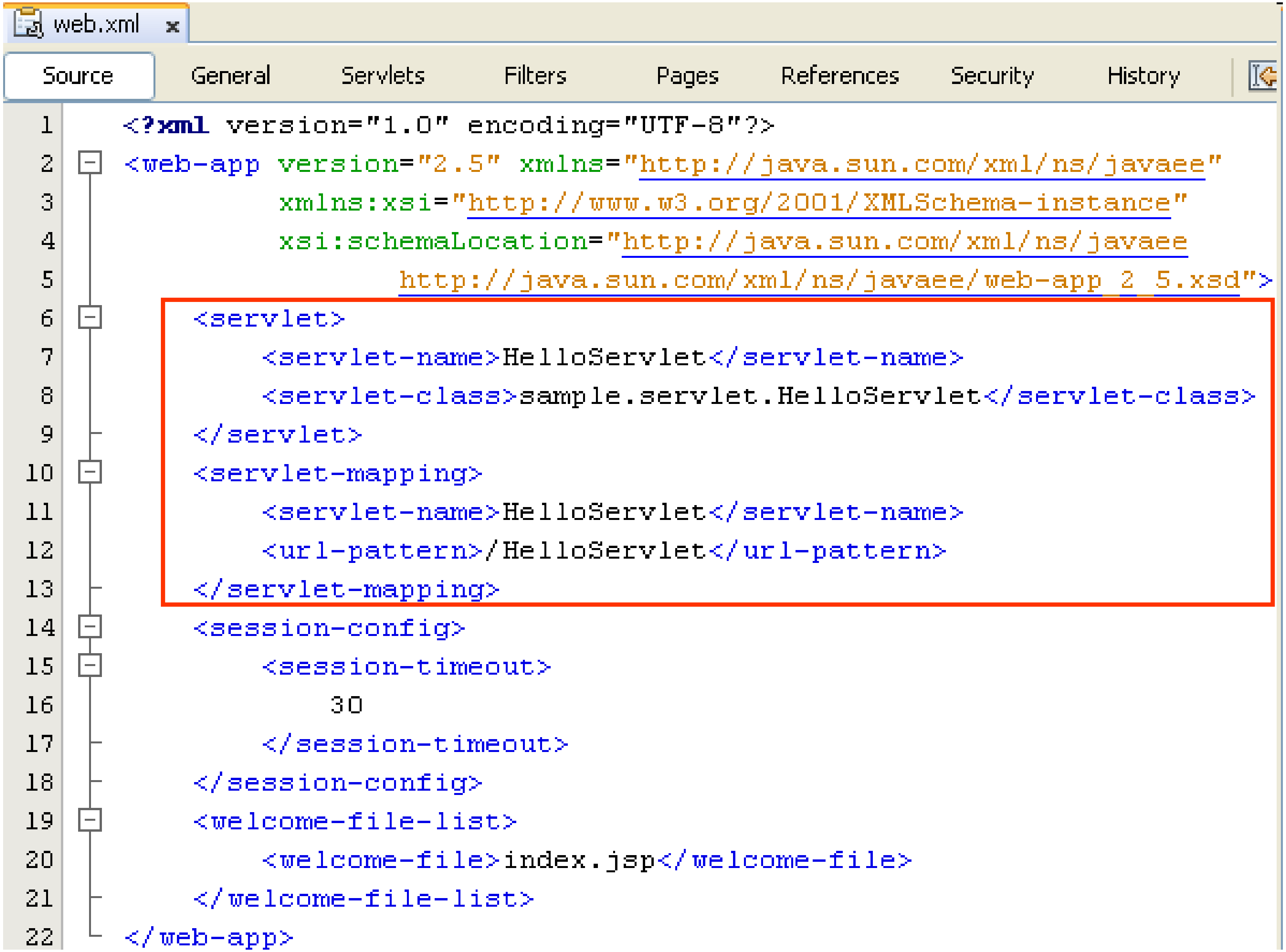
Create a Servlet



```
     *
     * @author Trong Khanh
     */
    public class HelloServlet extends HttpServlet {

        /**...*/
        protected void processRequest(HttpServletRequest request,
                HttpServletResponse response)
                throws ServletException, IOException {
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            try {
                /* TODO output your page here. You may use following s
                out.println("<!DOCTYPE html>");
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Hello</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("<h1>Welcome to Servlet World!!!</h1>");
                out.println("</body>");
                out.println("</html>");
            } finally {
                out.close();
            }
        }

        HttpServlet methods. Click on the + sign on the left to edit t
    }
```

Appendix

Create a Servlet



```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>sample.servlet.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/HelloServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

The Servlet Model

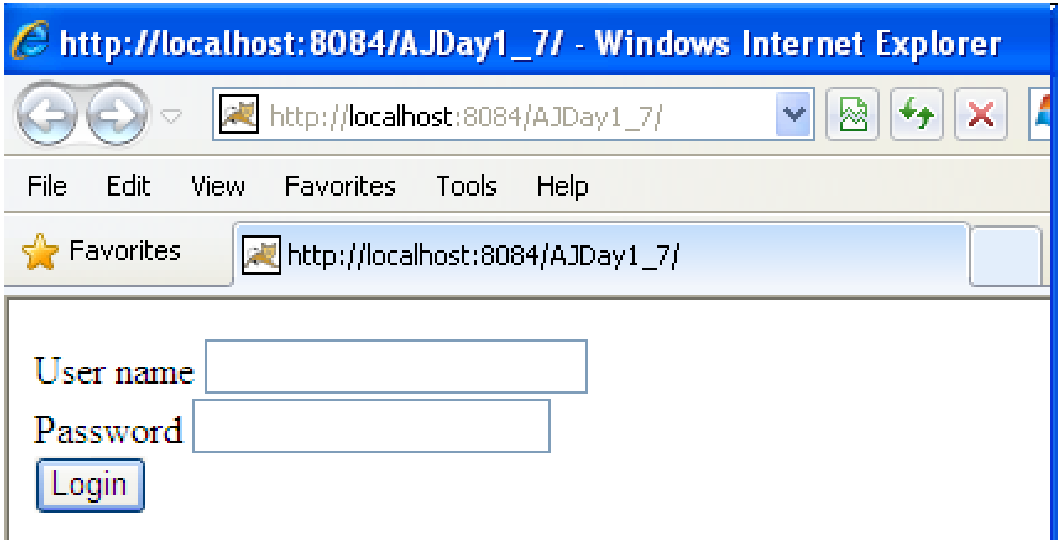
ServletRequest interface

- Provides **access** to **specific information about** the **request**
- Defines object (ServletRequest object)
  - **Containing actual request** (ex: protocol, URL, and type)
  - Containing **raw request** (ex: headers and input stream)
  - Containing client specific request **parameters**
  - Is **passed as an argument to** the **service**() method
- Some methods

Methods	Descriptions
getParameter	- **public String getParameter(String name)** - Returns the **value** of a specified parameter by the name (**or null or ""**) - String strUser = request.getParameter("txtUser");
getParameterNames	- **public Enumeration getParameterNames()** - Returns an **enumeration of string objects containing the name of parameters.** - Returns an **empty enumeration if** the request has **no parameters** - Enumeration strUser = request.getParameterName();
getParameterValues	- **public String[] getParameterValues(String names)** - Returns an **array of string objects** containing **all of the parameter values or null** if parameters do not exist. - String[] value = request.getParameterValues(”chkRemove”);

Appendix – The Servlet Model

HttpServletRequest interface – Examples


http://localhost:8084/AJDay1_7/ - Windows Internet Explorer
http://localhost:8084/AJDay1_7/
File
Edit
View
Favorites
Tools
Help
Favorites
http://localhost:8084/AJDay1_7/

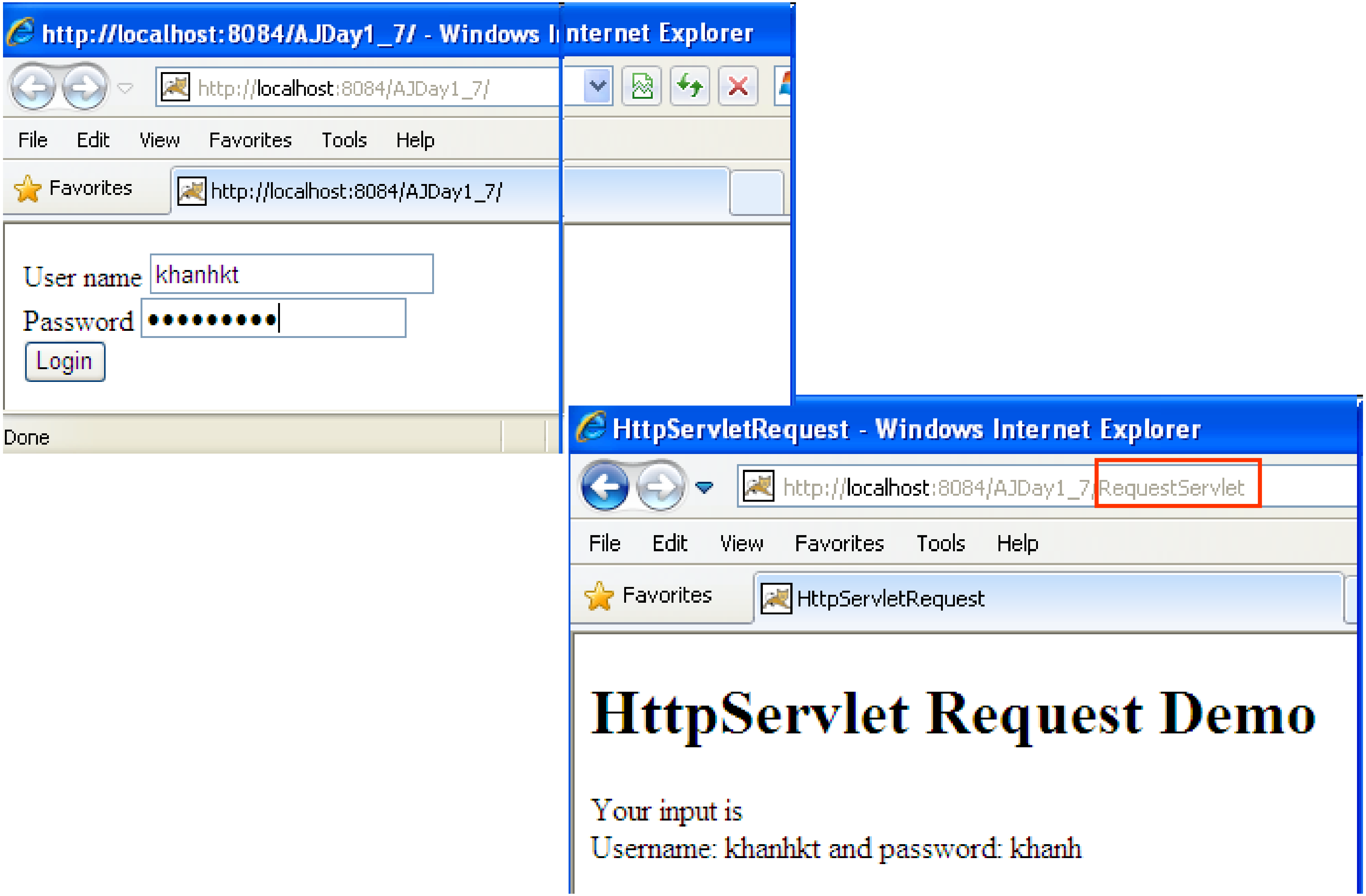
User name

Password

Login

Appendix – Servlet Model

HttpServletRequest interface – Examples


http://localhost:8084/AJDay1_7/ - Windows Internet Explorer
File
Edit
View
Favorites
Tools
Help
Favorites
Done
HttpServletRequest - Windows Internet Explorer
HttpServletRequest

http://localhost:8084/AJDay1_7/

User name khanhkt

Password ••••••••••

Login

http://localhost:8084/AJDay1_7/RequestServlet

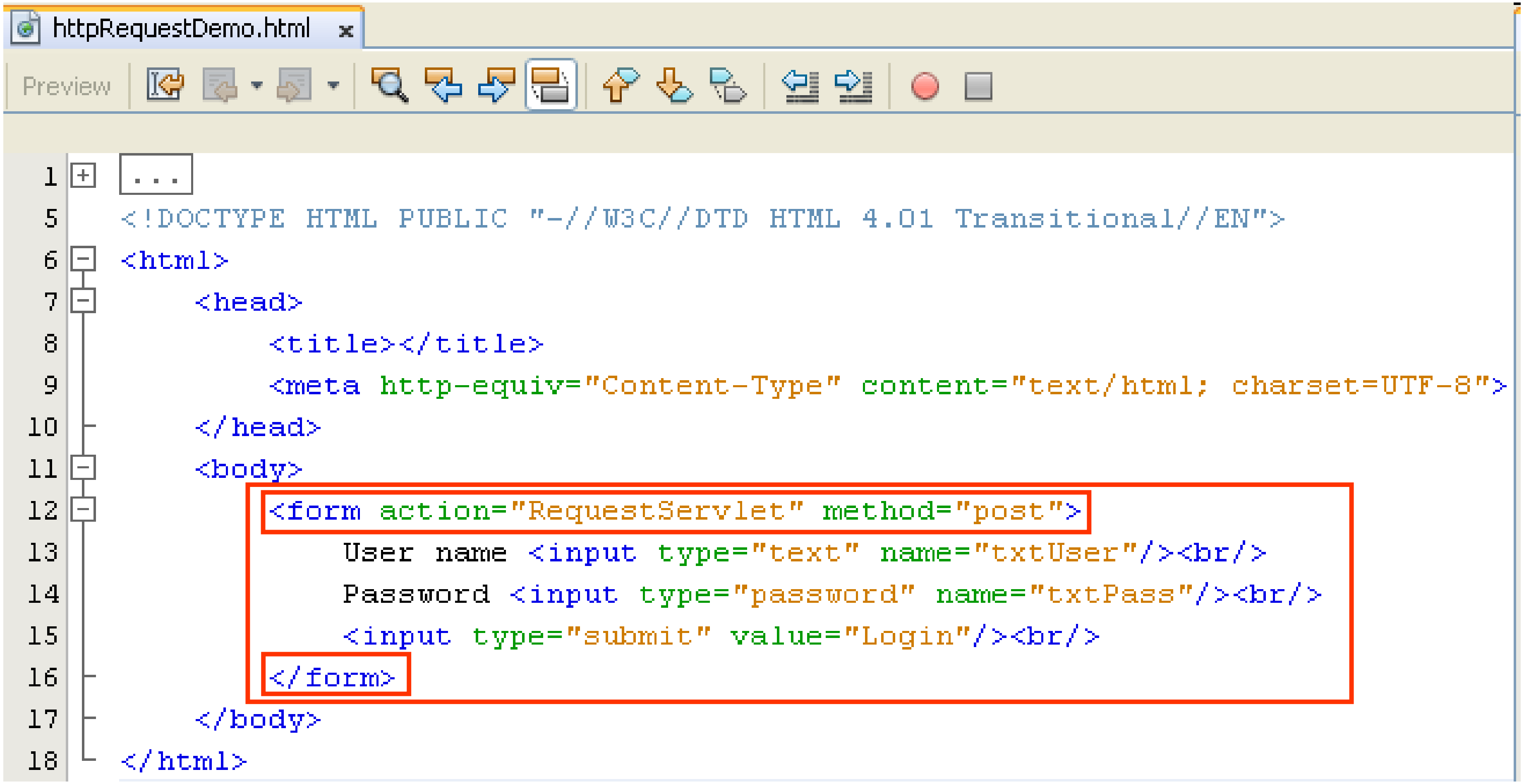
HttpServlet Request Demo

Your input is

Username: khanhkt and password: khanh

Appendix – The Servlet Model

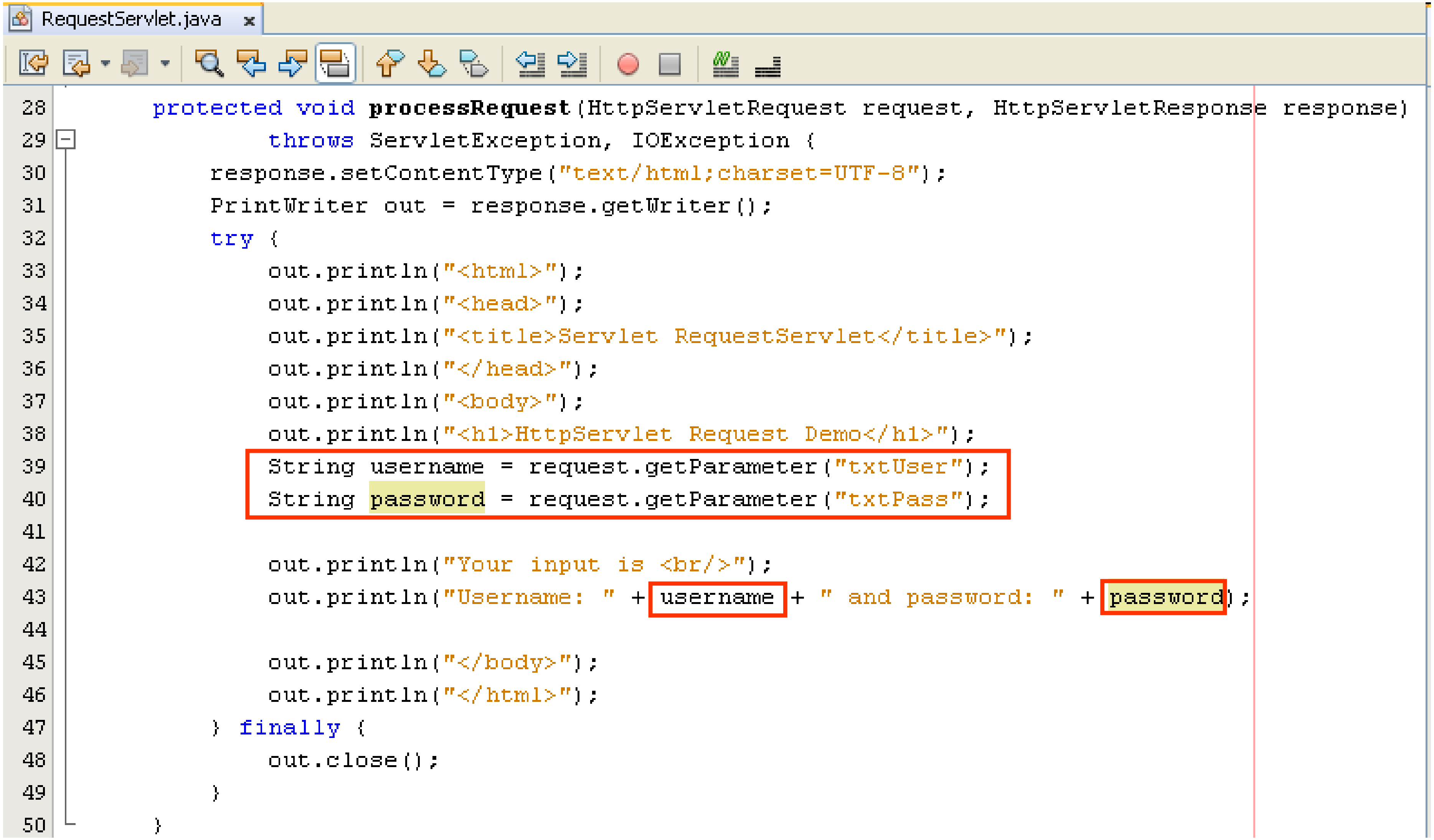
HttpServletRequest interface – Examples



```
...
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <form action="RequestServlet" method="post">
            User name <input type="text" name="txtUser"/><br/>
            Password <input type="password" name="txtPass"/><br/>
            <input type="submit" value="Login"/><br/>
        </form>
    </body>
</html>
```

Appendix – The Servlet Model

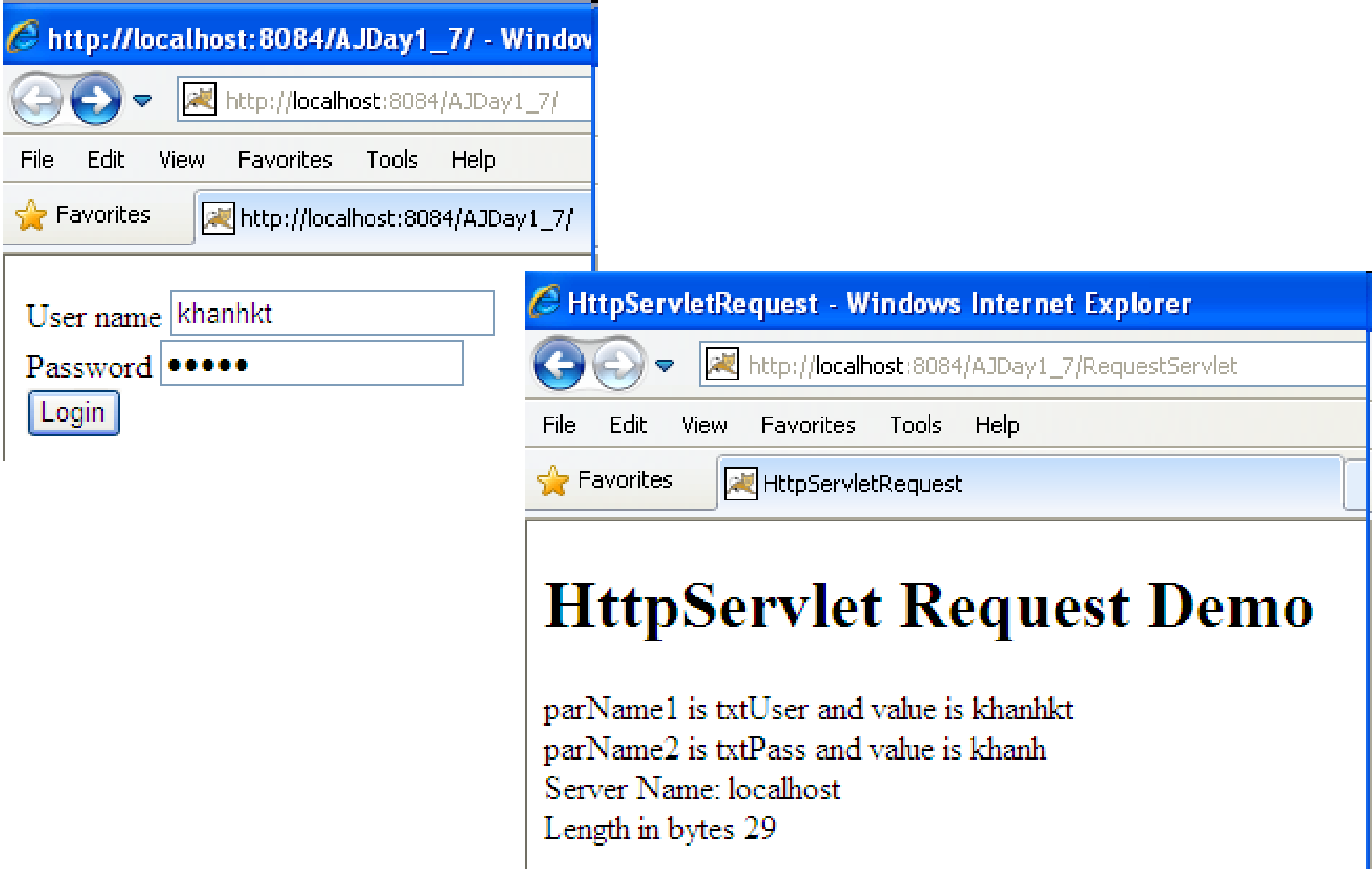
HttpServletRequest interface – Examples



```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet RequestServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>HttpServlet Request Demo</h1>");
        String username = request.getParameter("txtUser");
        String password = request.getParameter("txtPass");

        out.println("Your input is <br/>");
        out.println("Username: " + username + " and password: " + password);

        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples



http://localhost:8084/AJDay1_7/ - Window
http://localhost:8084/AJDay1_7/
File Edit View Favorites Tools Help
Favorites
http://localhost:8084/AJDay1_7/

User name khanhkt

Password ●●●●●

Login

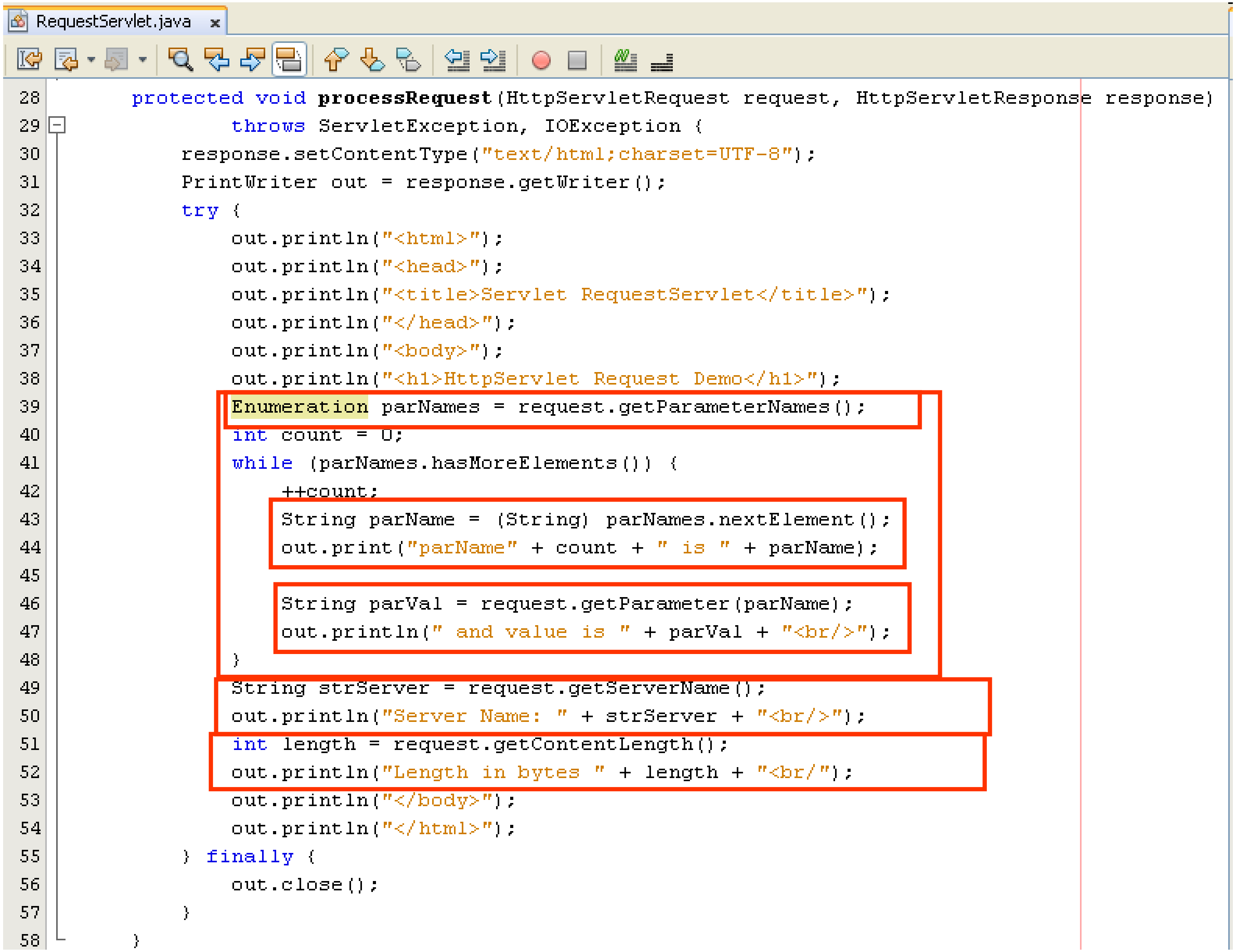
HttpServletRequest - Windows Internet Explorer
http://localhost:8084/AJDay1_7/RequestServlet
File Edit View Favorites Tools Help
Favorites
HttpServletRequest

HttpServlet Request Demo

parName1 is txtUser and value is khanhkt
parName2 is txtPass and value is khanh
Server Name: localhost
Length in bytes 29

Appendix – The Servlet Model

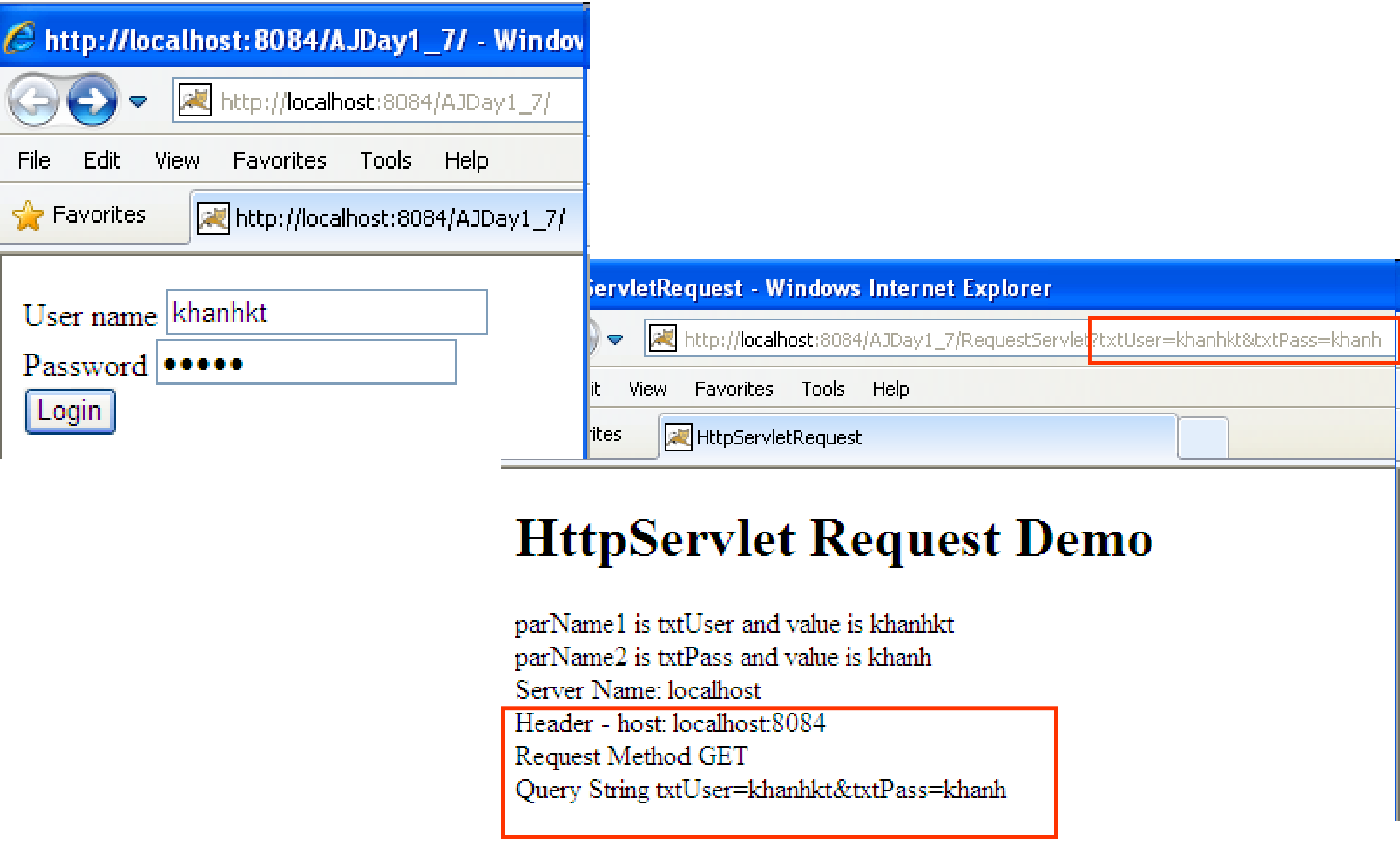
HttpServletRequest interface – Examples



```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet RequestServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>HttpServlet Request Demo</h1>");
        Enumeration parNames = request.getParameterNames();
        int count = 0;
        while (parNames.hasMoreElements()) {
            ++count;
            String parName = (String) parNames.nextElement();
            out.print("parName" + count + " is " + parName);

            String parVal = request.getParameter(parName);
            out.println(" and value is " + parVal + "<br/>");
        }
        String strServer = request.getServerName();
        out.println("Server Name: " + strServer + "<br/>");
        int length = request.getContentLength();
        out.println("Length in bytes " + length + "<br/");
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples


http://localhost:8084/AJDay1_7/ - Window
http://localhost:8084/AJDay1_7/
File
Edit
View
Favorites
Tools
Help
Favorites
http://localhost:8084/AJDay1_7/
Login
ServletRequest - Windows Internet Explorer
http://localhost:8084/AJDay1_7/RequestServlet?txtUser=khanhkt&txtPass=khanh
it
View
Favorites
Tools
Help
ites
HttpServletRequest

User name khanhkt

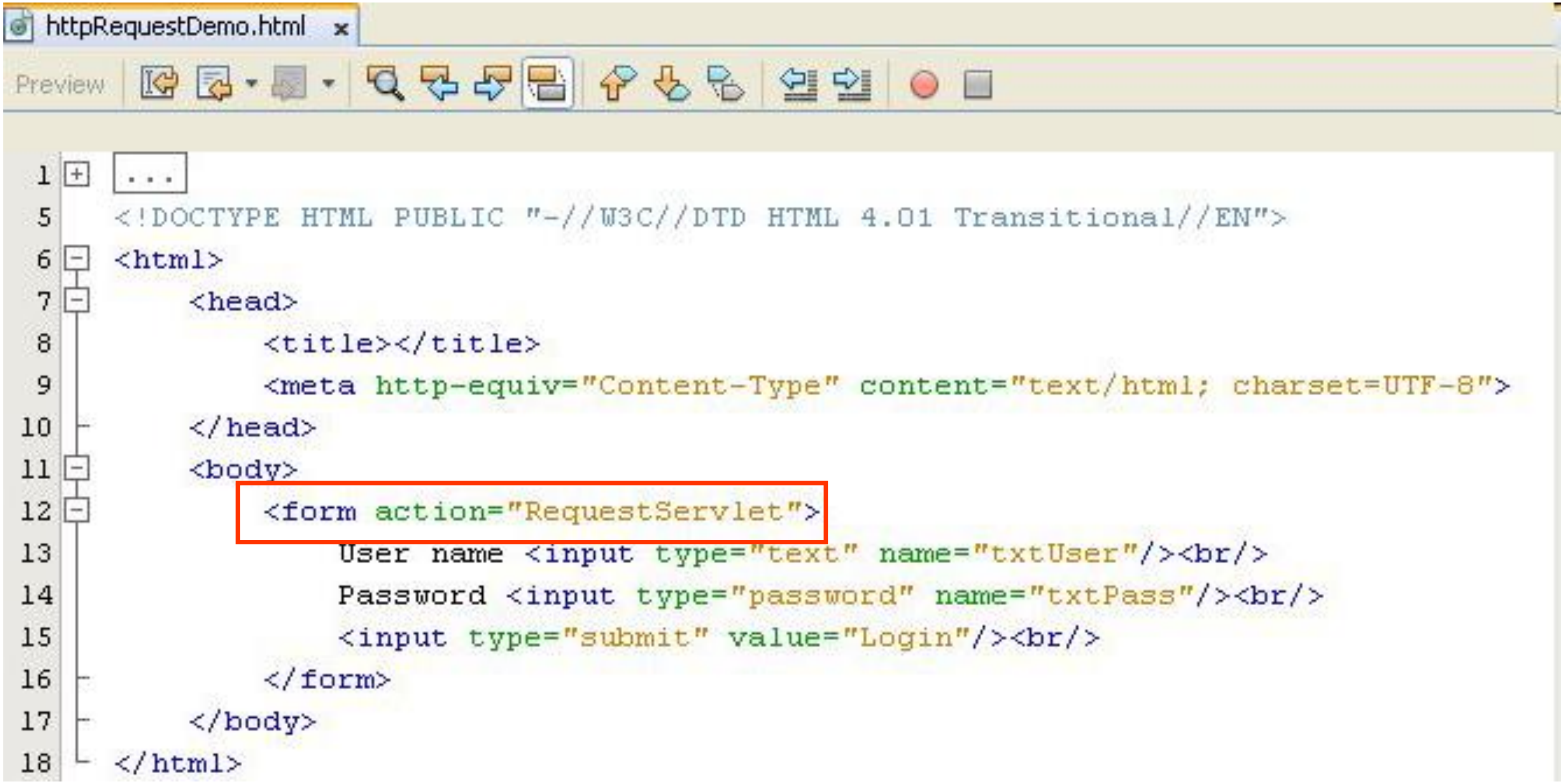
Password •••••

HttpServlet Request Demo

```
parName1 is txtUser and value is khanhkt
parName2 is txtPass and value is khanh
Server Name: localhost
Header - host: localhost:8084
Request Method GET
Query String txtUser=khanhkt&txtPass=khanh
```

Appendix – The Servlet Model

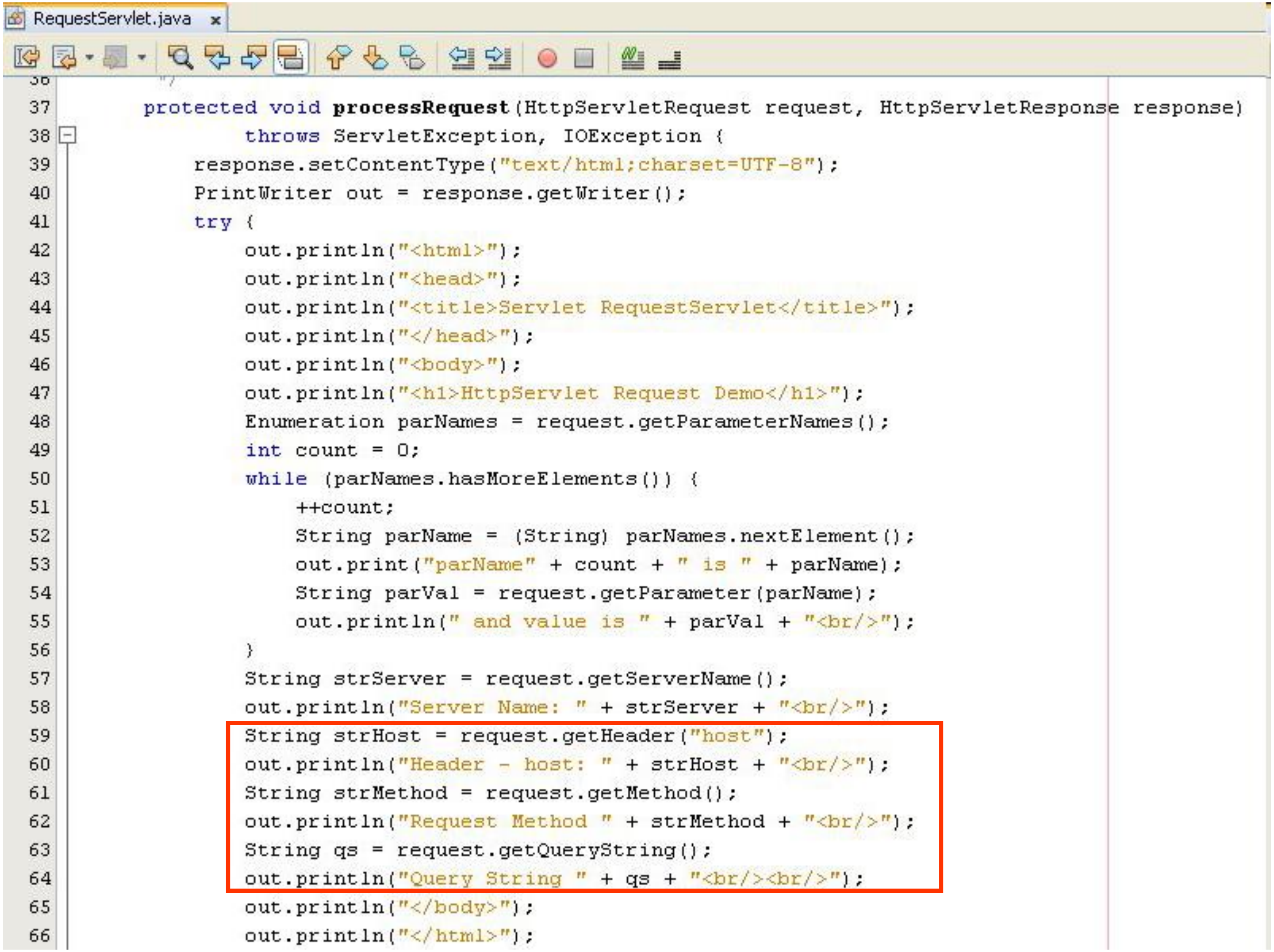
HttpServletRequest interface – Examples



```
...
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <form action="RequestServlet">
            User name <input type="text" name="txtUser"/><br/>
            Password <input type="password" name="txtPass"/><br/>
            <input type="submit" value="Login"/><br/>
        </form>
    </body>
</html>
```

Appendix – The Servlet Model

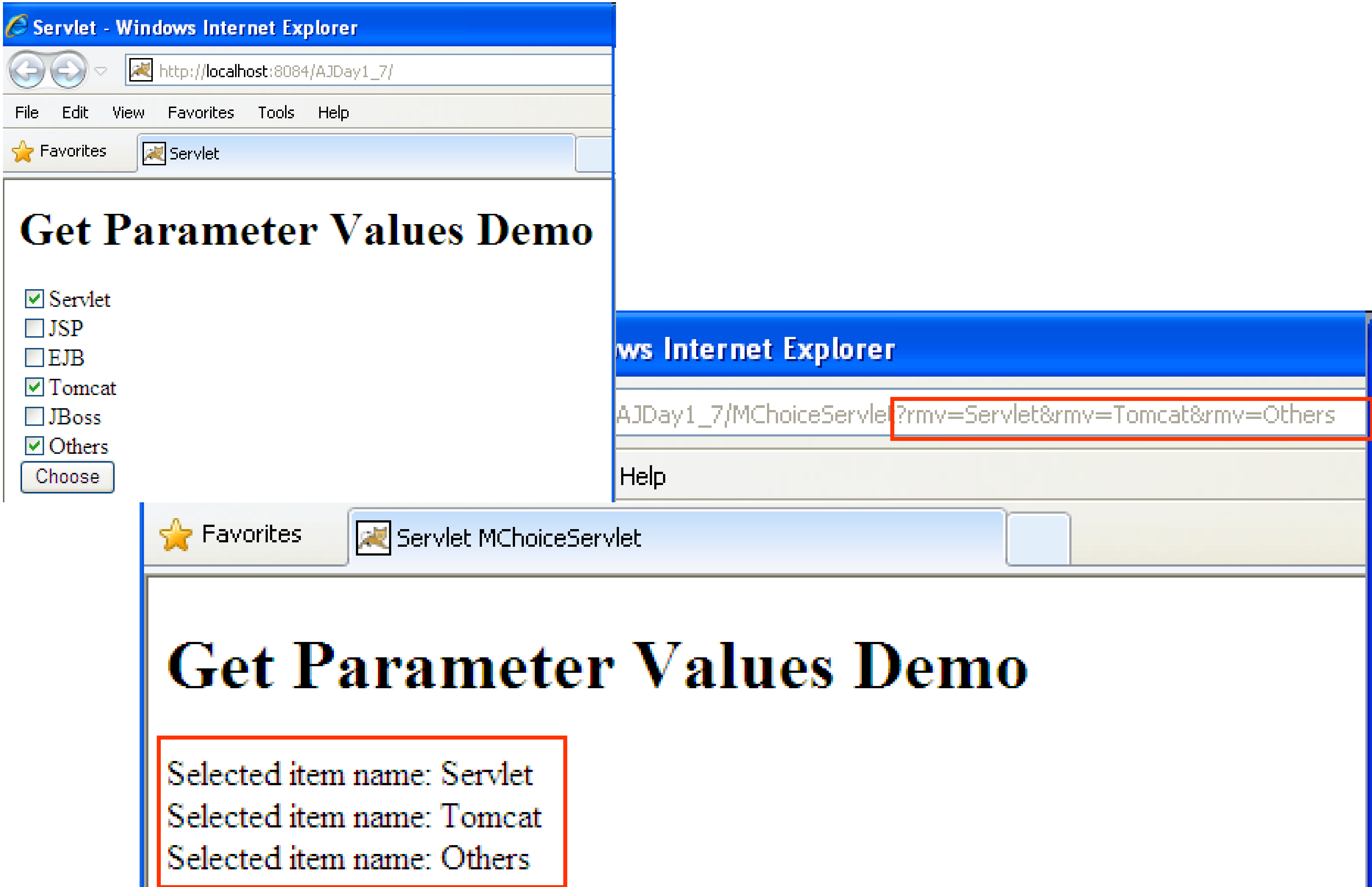
HttpServletRequest interface – Examples



```
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet RequestServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>HttpServlet Request Demo</h1>");
            Enumeration parNames = request.getParameterNames();
            int count = 0;
            while (parNames.hasMoreElements()) {
                ++count;
                String parName = (String) parNames.nextElement();
                out.print("parName" + count + " is " + parName);
                String parVal = request.getParameter(parName);
                out.println(" and value is " + parVal + "<br/>");
            }
            String strServer = request.getServerName();
            out.println("Server Name: " + strServer + "<br/>");
            String strHost = request.getHeader("host");
            out.println("Header - host: " + strHost + "<br/>");
            String strMethod = request.getMethod();
            out.println("Request Method " + strMethod + "<br/>");
            String qs = request.getQueryString();
            out.println("Query String " + qs + "<br/><br/>");
            out.println("</body>");
            out.println("</html>");
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples



http://localhost:8084/AJDay1_7/

Get Parameter Values Demo

☑ Servlet
□ JSP
□ EJB
☑ Tomcat
□ JBoss
☑ Others

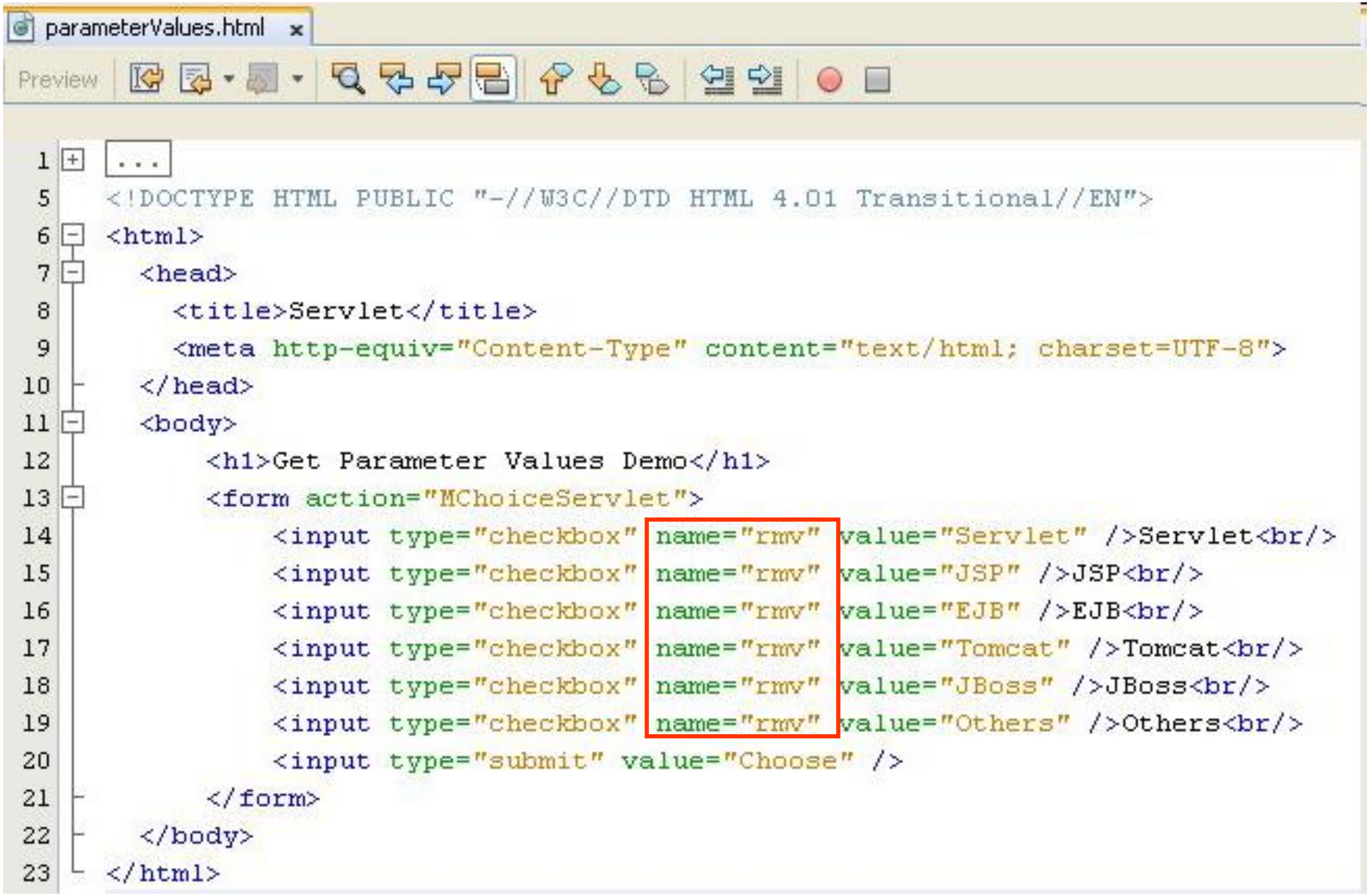
AJDay1_7/MChoiceServlet?rmv=Servlet&rmv=Tomcat&rmv=Others

Get Parameter Values Demo

Selected item name: Servlet
Selected item name: Tomcat
Selected item name: Others

Appendix – The Servlet Model

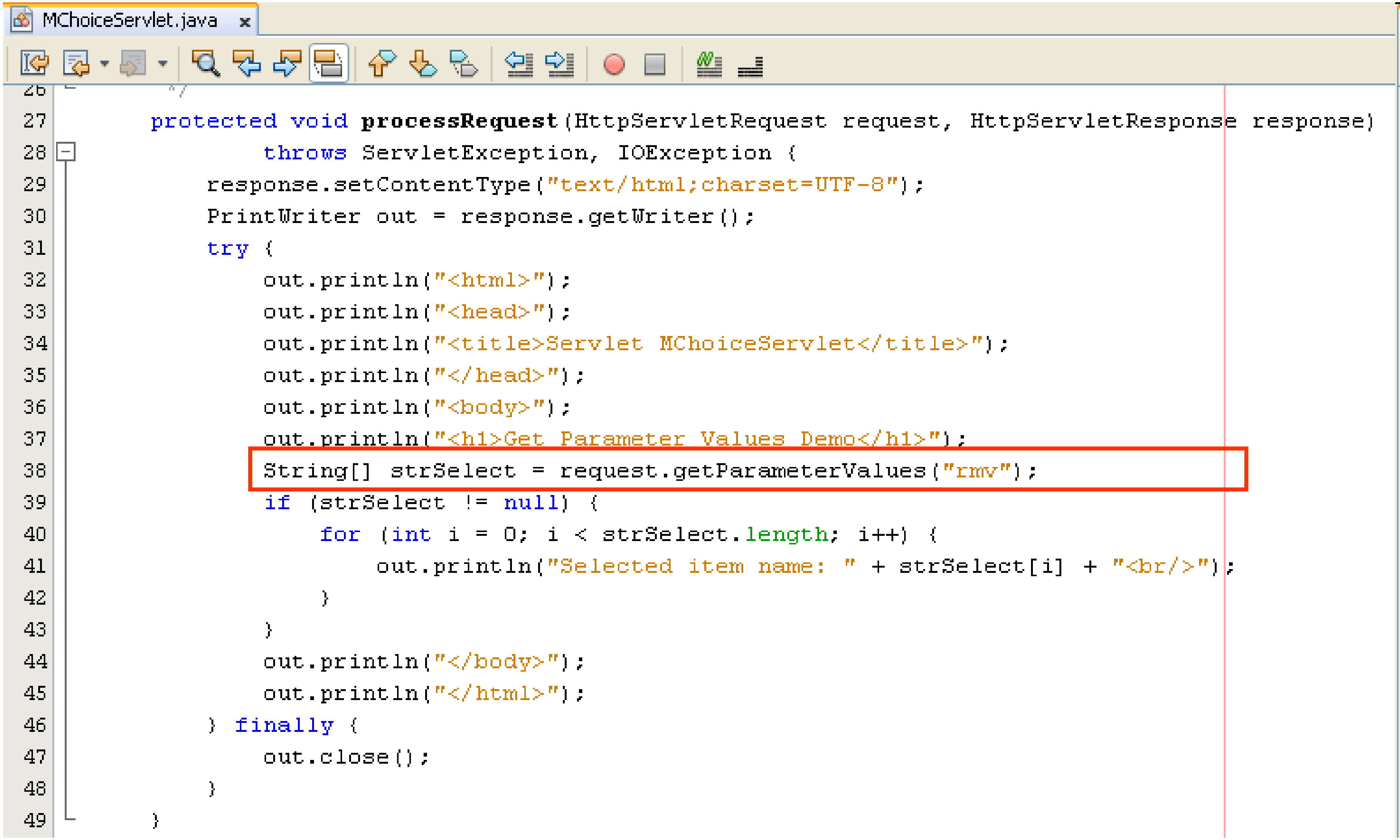
HttpServletRequest interface – Examples



```
...
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Servlet</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
      <h1>Get Parameter Values Demo</h1>
      <form action="MChoiceServlet">
          <input type="checkbox" name="rmv" value="Servlet" />Servlet<br/>
          <input type="checkbox" name="rmv" value="JSP" />JSP<br/>
          <input type="checkbox" name="rmv" value="EJB" />EJB<br/>
          <input type="checkbox" name="rmv" value="Tomcat" />Tomcat<br/>
          <input type="checkbox" name="rmv" value="JBoss" />JBoss<br/>
          <input type="checkbox" name="rmv" value="Others" />Others<br/>
          <input type="submit" value="Choose" />
      </form>
  </body>
</html>
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples

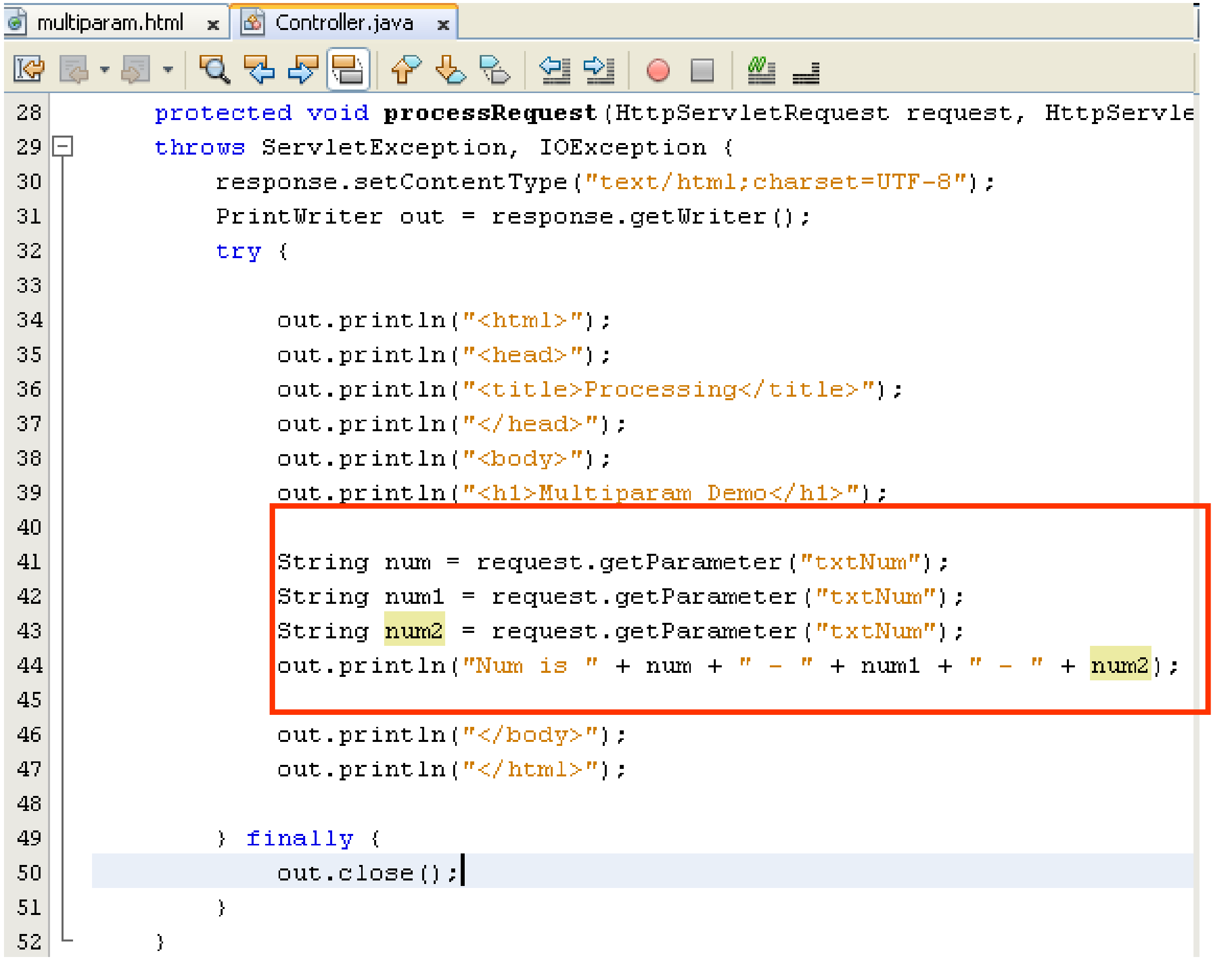


```
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet MChoiceServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Get Parameter Values Demo</h1>");
            String[] strSelect = request.getParameterValues("rmv");
            if (strSelect != null) {
                for (int i = 0; i < strSelect.length; i++) {
                    out.println("Selected item name: " + strSelect[i] + "<br/>");
                }
            }
            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples

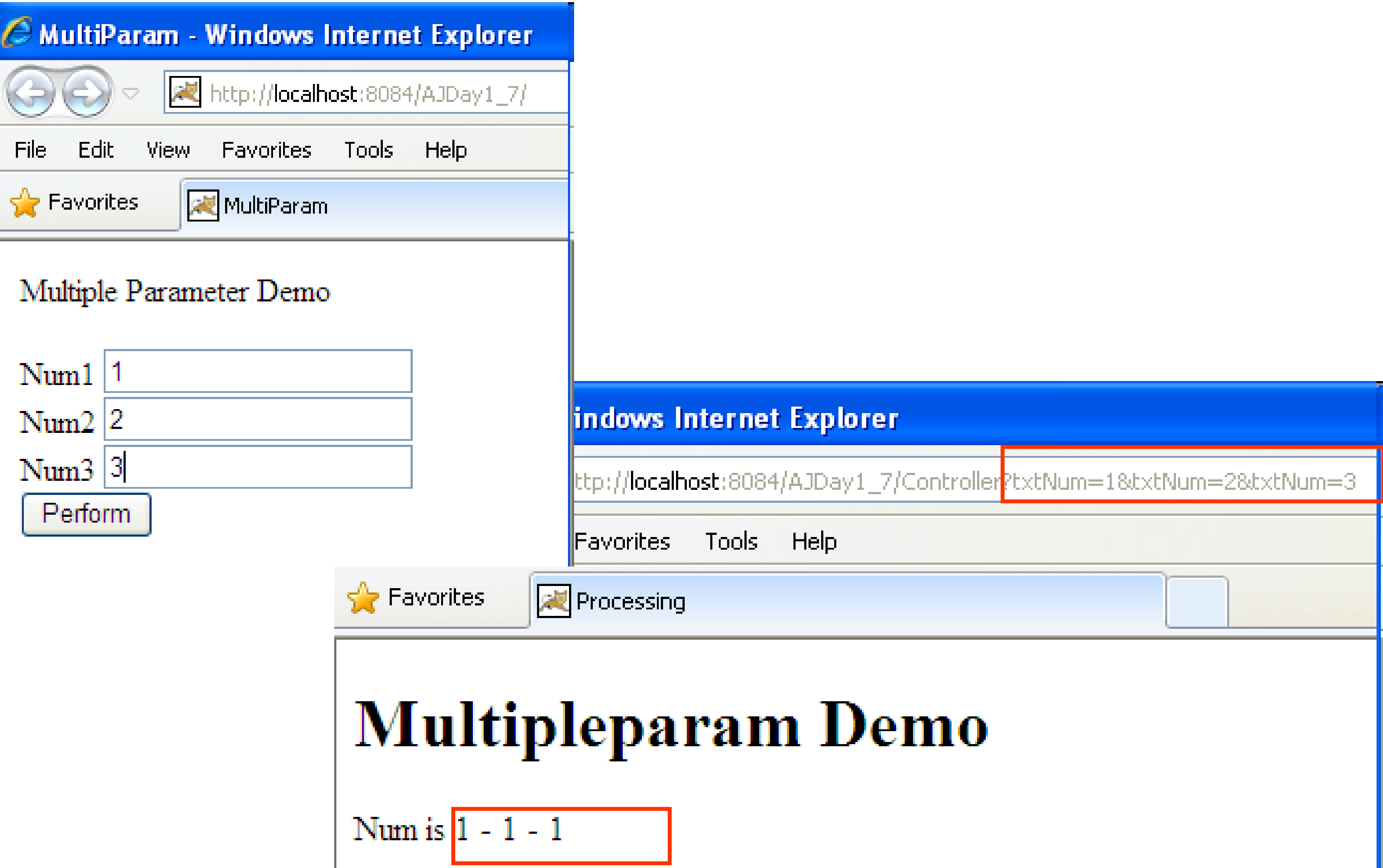
```
<body>
    <form action="Controller">
        Num1 <input type="text" name="txtNum"/> <br/>
        Num2 <input type="text" name="txtNum"/> <br/>
        Num3 <input type="text" name="txtNum"/> <br/>
        <input type="submit" value="Perform" />
    </form>

</body>
```



```
    protected void processRequest(HttpServletRequest request, HttpServle
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {

            out.println("<html>");
            out.println("<head>");
            out.println("<title>Processing</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Multiparam Demo</h1>");

            String num = request.getParameter("txtNum");
            String num1 = request.getParameter("txtNum");
            String num2 = request.getParameter("txtNum");
            out.println("Num is " + num + " - " + num1 + " - " + num2);

            out.println("</body>");
            out.println("</html>");

        } finally {
            out.close();
        }
    }
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples



http://localhost:8084/AJDay1_7/

Multiple Parameter Demo

Num1 1
Num2 2
Num3 3

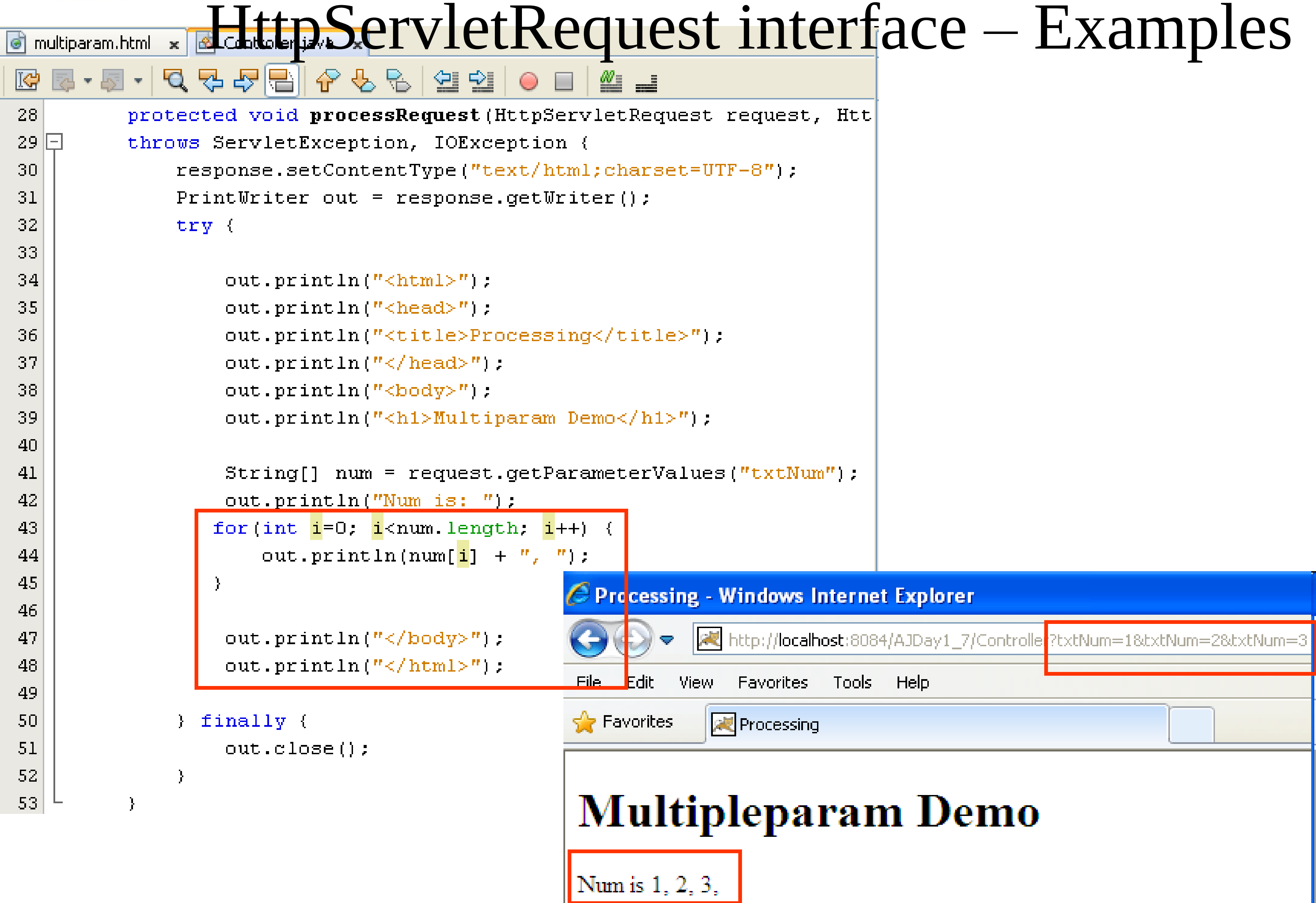
ttp://localhost:8084/AJDay1_7/Controller?txtNum=1&txtNum=2&txtNum=3

Multipleparam Demo

Num is 1 - 1 - 1

Appendix – The Servlet Model

HttpServletRequest interface – Examples



```
        protected void processRequest(HttpServletRequest request, Htt
        throws ServletException, IOException {
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            try {

                out.println("<html>");
                out.println("<head>");
                out.println("<title>Processing</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("<h1>Multiparam Demo</h1>");

                String[] num = request.getParameterValues("txtNum");
                out.println("Num is: ");
               for(int i=0; i<num.length; i++) {
                   out.println(num[i] + ", ");
               }

                out.println("</body>");
                out.println("</html>");

            } finally {
                out.close();
            }
        }
```

http://localhost:8084/AJDay1_7/Controller?txtNum=1&txtNum=2&txtNum=3

Multipleparam Demo

Num is 1, 2, 3,

The Servlet Model

ServletResponse interface

- Is **response sent** by the servlet to the **client**
- Include **all the methods** needed **to create and manipulate** a **servlet's output**
- **Retrieve** an **output stream** to send data to the client, **decide** on the **content type** ...
- **Define objects** passed as an argument to service() method
- Some methods

Methods	Descriptions
getContentType	- **public String getContentType()** - Returns the **Multipurpose Internet Mail Extensions** (MIME) type of the request body or **null** if the type is not known - String contentType = response.getContentType();
getWriter	- **public PrintWriter getWriter() throws IOException** - Returns **an object of PrintWriter** class that **sends character text to the client, particular Browser.** - PrintWriter out = response.getWriter();

The Servlet Model

ServletResponse interface

Methods	Descriptions
getOutputStream	- **public ServletOutputStream getOutputStream() throws IOException** - Uses ServletOutputStream object to **write response as binary data to the client.** - ServletOutputStream out = response.getOutputStream(); - 02 supporting methods + **public void print(boolean b) throws IOException** . **writes a boolean value** to the client with no carriage return line feed (CRLF) character at the end . out.print(b); + **public void println(char c) throws IOException** . same as the print methods but it **writes a character value** to the client, followed by a carriage return line feed (CRLF)
setContentType	- **public void setContentType(String str)** - Used to **set format in which the data** is **sent to the client**, either normal text formate or html format - **Ex**: response.setContentType(“text/html”);

The Servlet Model

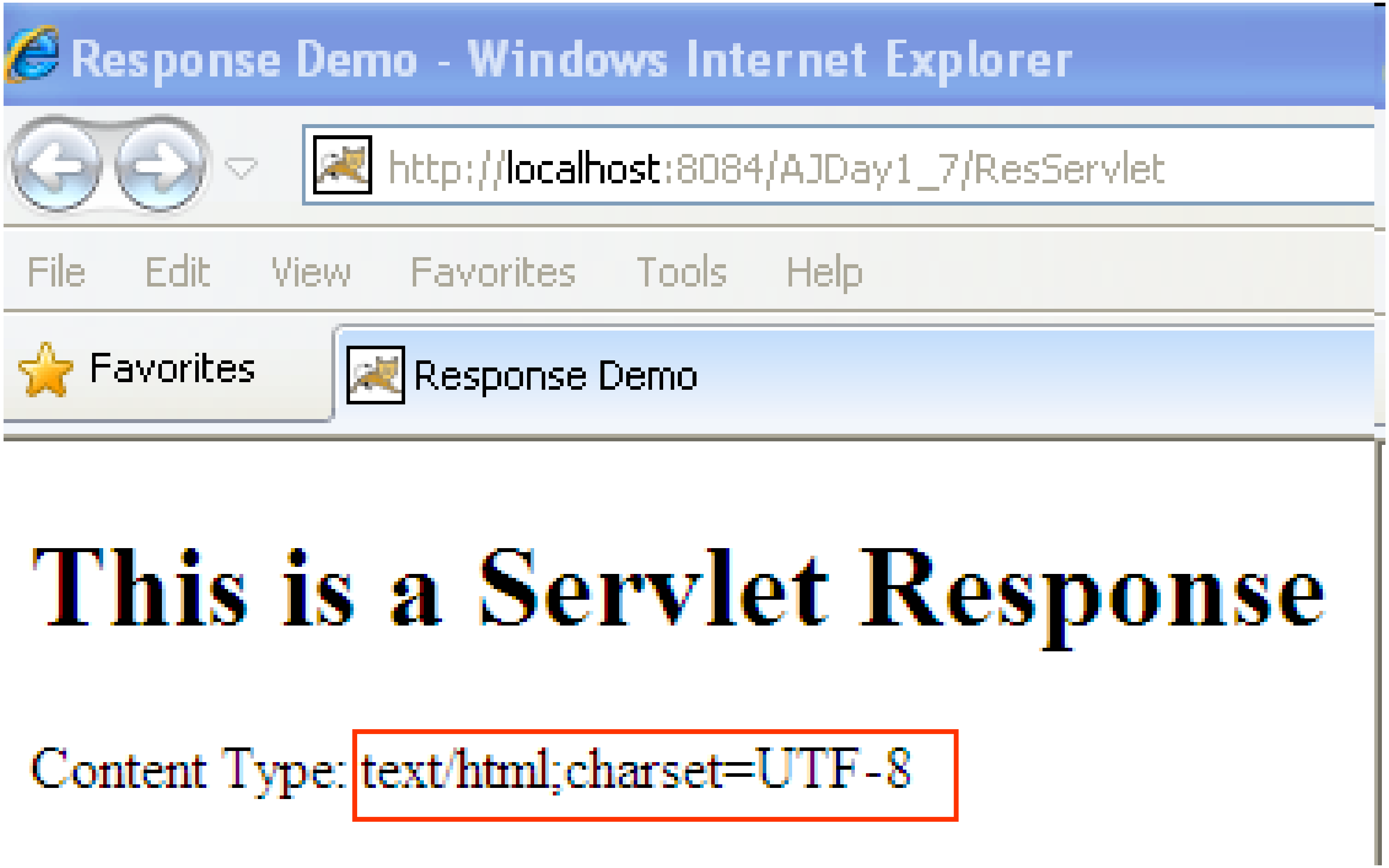
HttpServletResponse interface

- **Extends ServletResponse Interface**
- **Defines HttpServlet objects** to **pass** as an argument **to the service**() method to the client
- Set HTTP response, HTTP header, set content type of the response, acquire a text stream for the response, acquire a binary stream for the response, redirect an HTTP request to another URL or add cookies to the response

Methods	Descriptions
encodeRedirectURL	- **public String encodeRedirectURL (String url)** - **Encodes** the **specified URL for use** in the **sendRedirect** method, or **if encoding is not needed**, returns the URL unchanged
sendRedirect	- **public void sendRedirect(String URL) throws IOException** - Sends a redirect response to the client using the **specified redirect location URL** - the servlet using the sendRedirect method to decide the request handled by particular servlet or - Ex: response.sendRedirect(“process.jsp”);

Appendix – The Servlet Model

HttpServletResponse interface - Example


Response Demo - Windows Internet Explorer
http://localhost:8084/AJDay1_7/ResServlet
File
Edit
View
Favorites
Tools
Help
Favorites
Response Demo

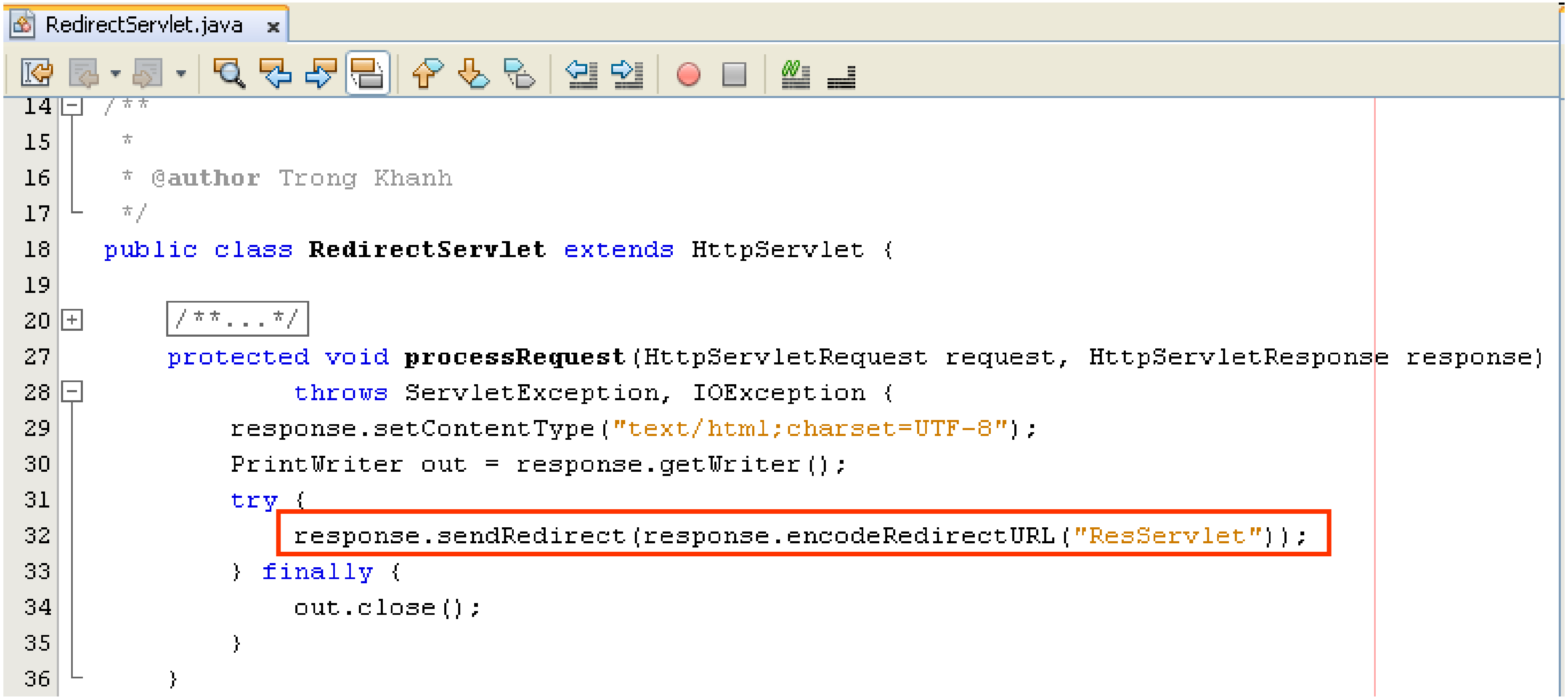
This is a Servlet Response

Content Type: text/html;charset=UTF-8

Appendix – The Servlet Model

HttpServletResponse interface - Example

- Using sendRedirect

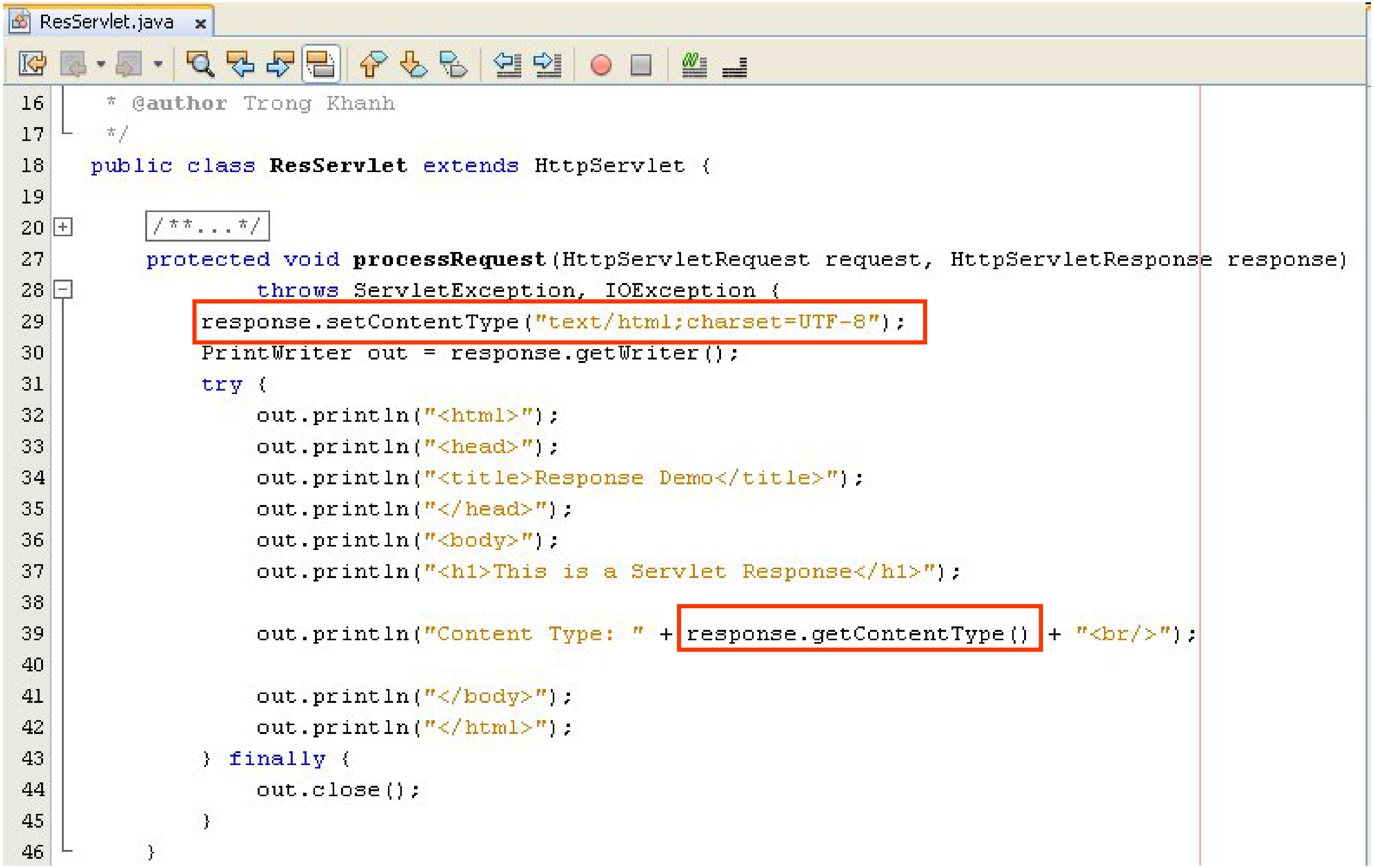


```
/**
 *
 * @author Trong Khanh
 */
public class RedirectServlet extends HttpServlet {

    /**...*/
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            response.sendRedirect(response.encodeRedirectURL("ResServlet"));
        } finally {
            out.close();
        }
    }
```

Appendix – The Servlet Model

HttpServletResponse interface - Example

- ResServlet



```
 * @author Trong Khanh
 */
public class ResServlet extends HttpServlet {

    /**...*/
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Response Demo</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>This is a Servlet Response</h1>");

            out.println("Content Type: " + response.getContentType() + "<br/>");

            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }
```

The Servlet Model

HttpServlet class

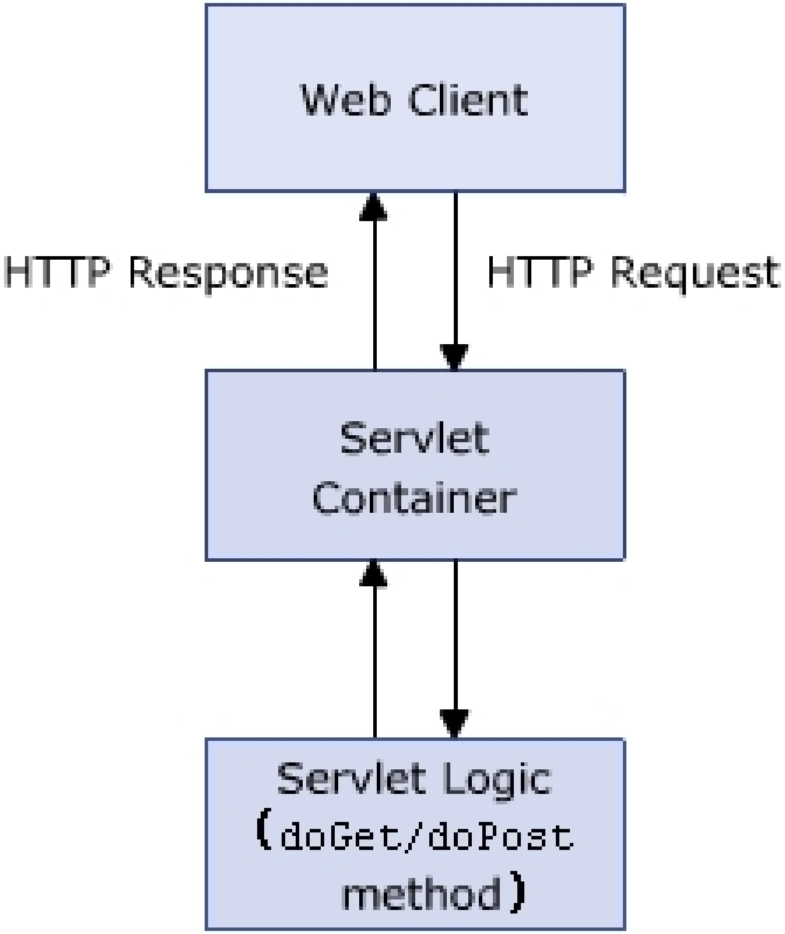
- The protocol **defines** a set of **text-based request messages** called HTTP ‘methods’ **implemented in *HttpServlet* class**
- Provides **an abstract class** to **create** an **HTTP Servlet**
- **Extends** the **GenericServlet class**
- A subclass of HttpServlet class **must override at least** one of the following methods: **doGet(), doPost,** doPut(), doDelete(), init(), destroy(), and getServletInfo
- Some methods to process the request

Methods	**Descriptions**
doGet	- **protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException** - **called by container** to **handle** the **GET** request. - This method is **called through service() method**
doPost	- **protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException** - **called** by **container** to **handle** the **POST request.** - This method is **called through service() method**

The Servlet Model

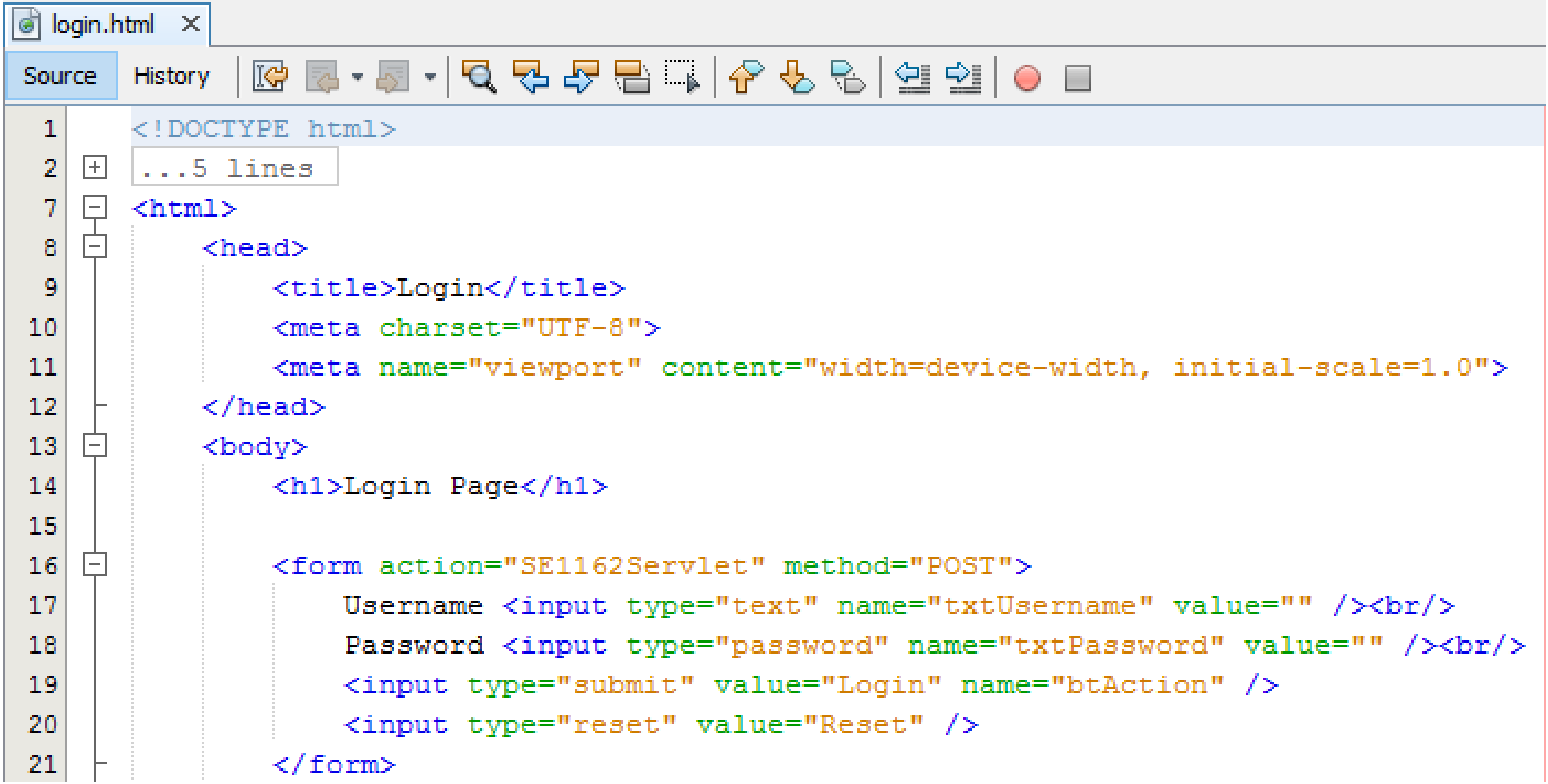
HttpServletRequest interface

- **Extends ServletRequest** Interface
- **Add a few more methods** for handling HTTP-specific request data
- **Defines an HttpServletRequest object passed as an argument to the service**() method



Appendix – Build The Simple Web

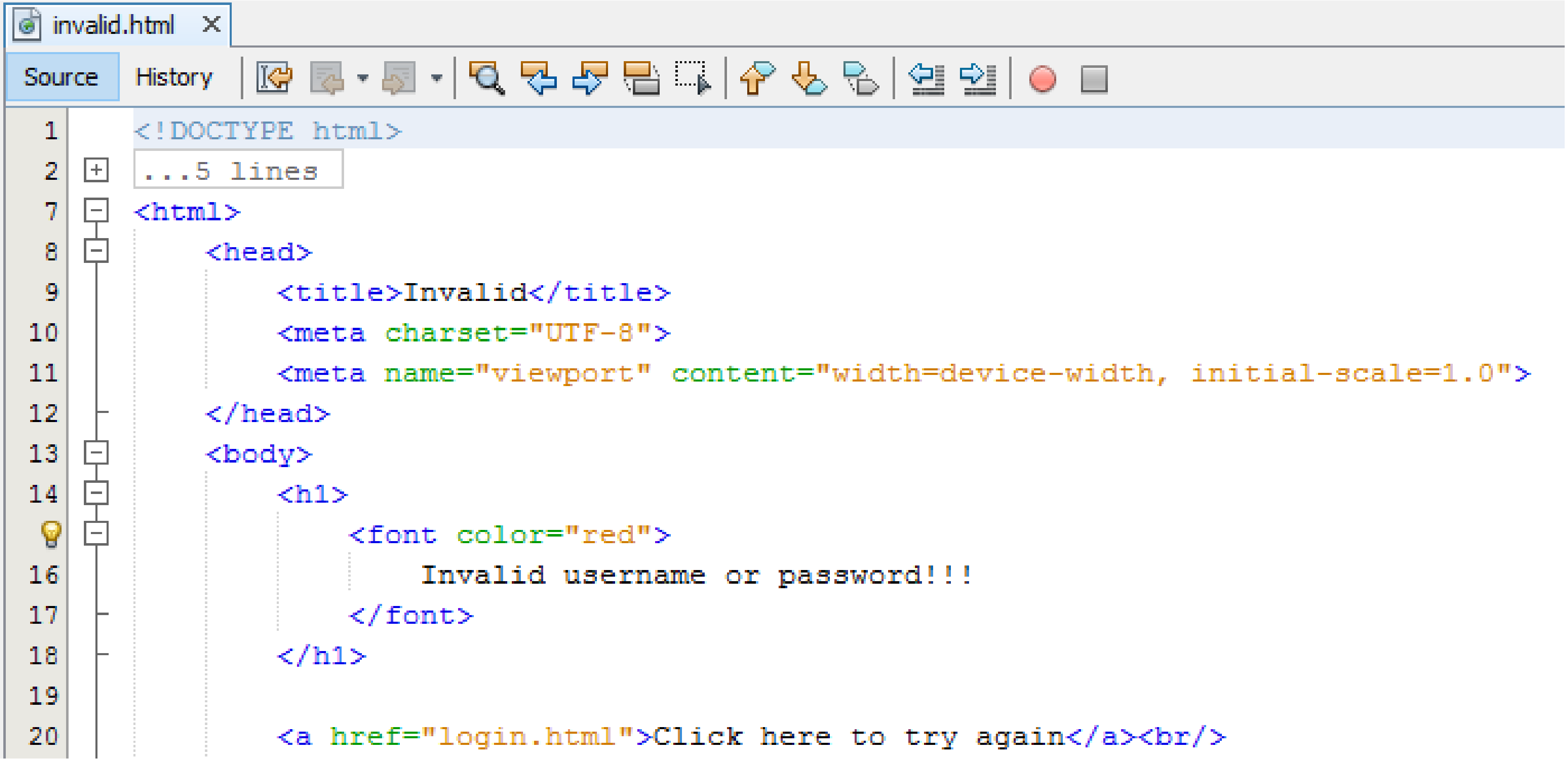
Login Page



```
<!DOCTYPE html>
...5 lines
<html>
    <head>
        <title>Login</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Login Page</h1>

        <form action="SE1162Servlet" method="POST">
            Username <input type="text" name="txtUsername" value="" /><br/>
            Password <input type="password" name="txtPassword" value="" /><br/>
            <input type="submit" value="Login" name="btAction" />
            <input type="reset" value="Reset" />
        </form>
```

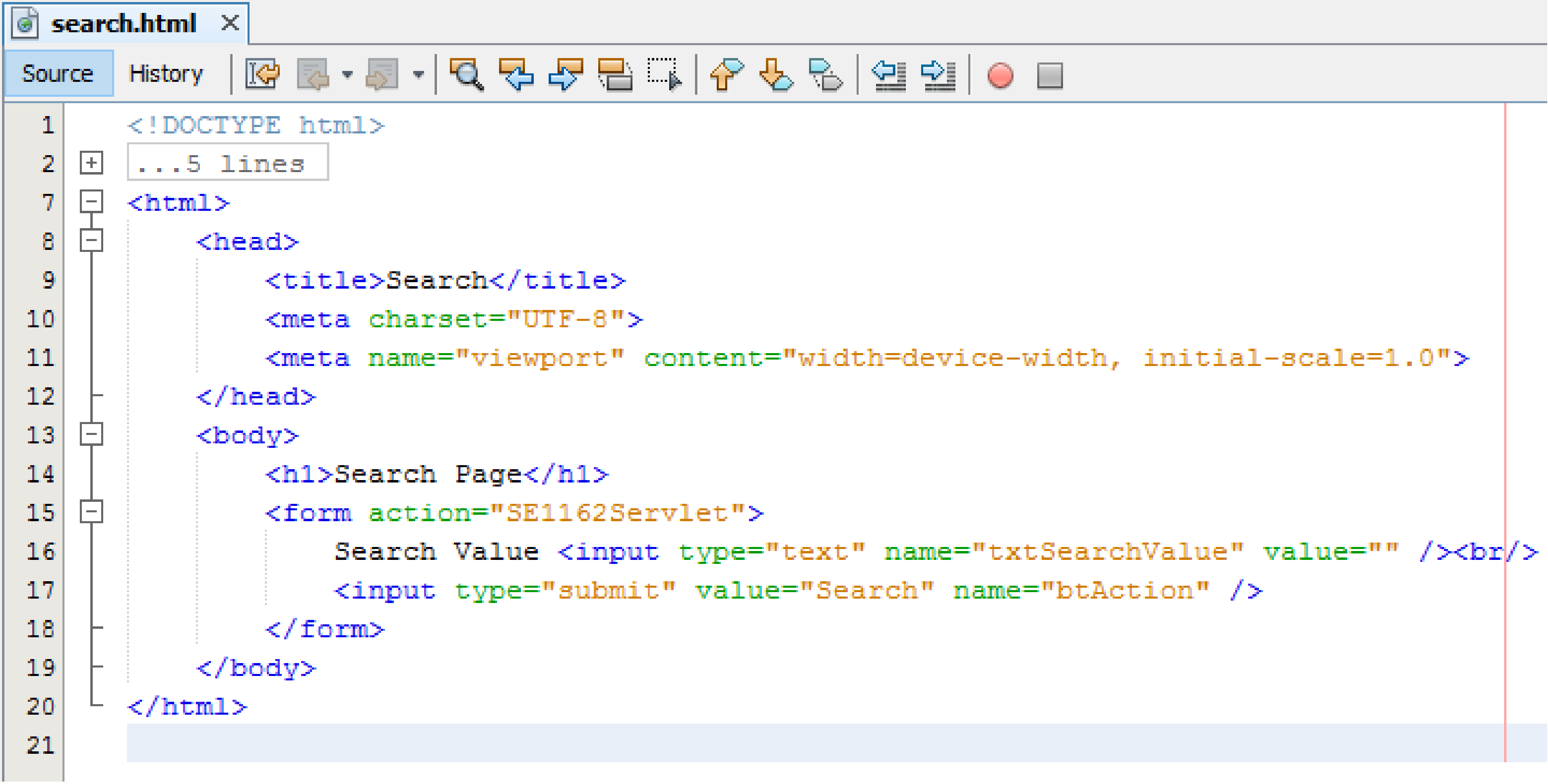
Appendix – Build The Simple Web

Invalid Page



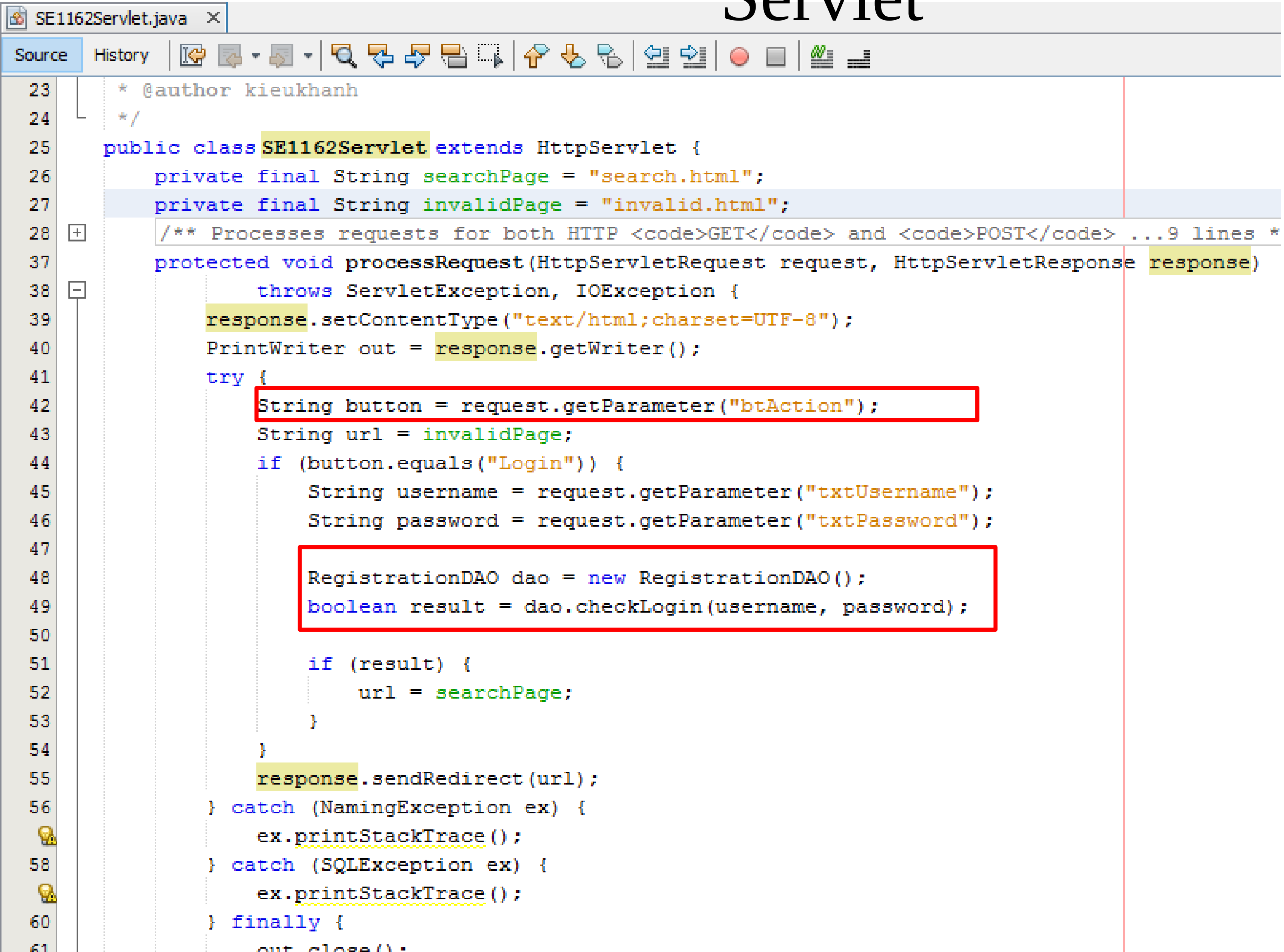
```
<!DOCTYPE html>
...5 lines
<html>
    <head>
        <title>Invalid</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>
            <font color="red">
                Invalid username or password!!!
            </font>
        </h1>

        <a href="login.html">Click here to try again</a><br/>
```

Appendix – Build The Simple Web Search Page



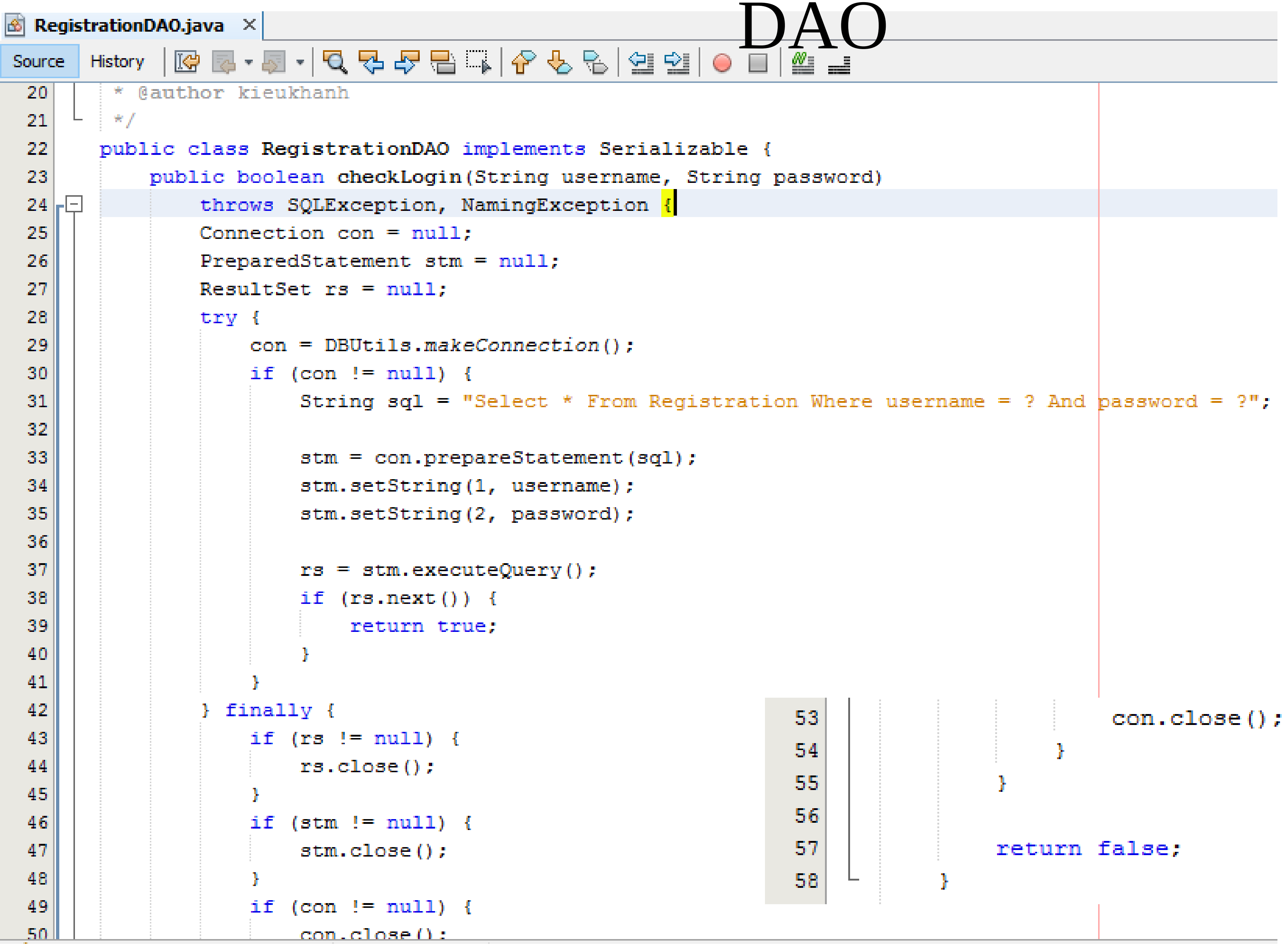
```
<!DOCTYPE html>
...5 lines
<html>
    <head>
        <title>Search</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Search Page</h1>
        <form action="SE1162Servlet">
            Search Value <input type="text" name="txtSearchValue" value="" /><br/>
            <input type="submit" value="Search" name="btAction" />
        </form>
    </body>
</html>

```

Appendix – Build The Simple Web Servlet



```
 * @author kieukhanh
 */
public class SE1162Servlet extends HttpServlet {
    private final String searchPage = "search.html";
    private final String invalidPage = "invalid.html";
    /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines *
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String button = request.getParameter("btAction");
            String url = invalidPage;
            if (button.equals("Login")) {
                String username = request.getParameter("txtUsername");
                String password = request.getParameter("txtPassword");

                RegistrationDAO dao = new RegistrationDAO();
                boolean result = dao.checkLogin(username, password);

                if (result) {
                    url = searchPage;
                }
            }
            response.sendRedirect(url);
        } catch (NamingException ex) {
            ex.printStackTrace();
        } catch (SQLException ex) {
            ex.printStackTrace();
        } finally {
            out.close();
```

Appendix – Build The Simple Web DAO



```java
 * @author kieukhanh
 */
public class RegistrationDAO implements Serializable {
    public boolean checkLogin(String username, String password)
        throws SQLException, NamingException {
        Connection con = null;
        PreparedStatement stm = null;
        ResultSet rs = null;
        try {
            con = DBUtils.makeConnection();
            if (con != null) {
                String sql = "Select * From Registration Where username = ? And password = ?";

                stm = con.prepareStatement(sql);
                stm.setString(1, username);
                stm.setString(2, password);

                rs = stm.executeQuery();
                if (rs.next()) {
                    return true;
                }
            }
        } finally {
            if (rs != null) {
                rs.close();
            }
            if (stm != null) {
                stm.close();
            }
            if (con != null) {
                con.close();
```

```java
                con.close();
            }
        }

        return false;
    }
```