

The Servlet Model

HTTP Methods

Form Parameters

Requests

Responses

Servlet Life Cycle

#Servlet #Video_Servlet

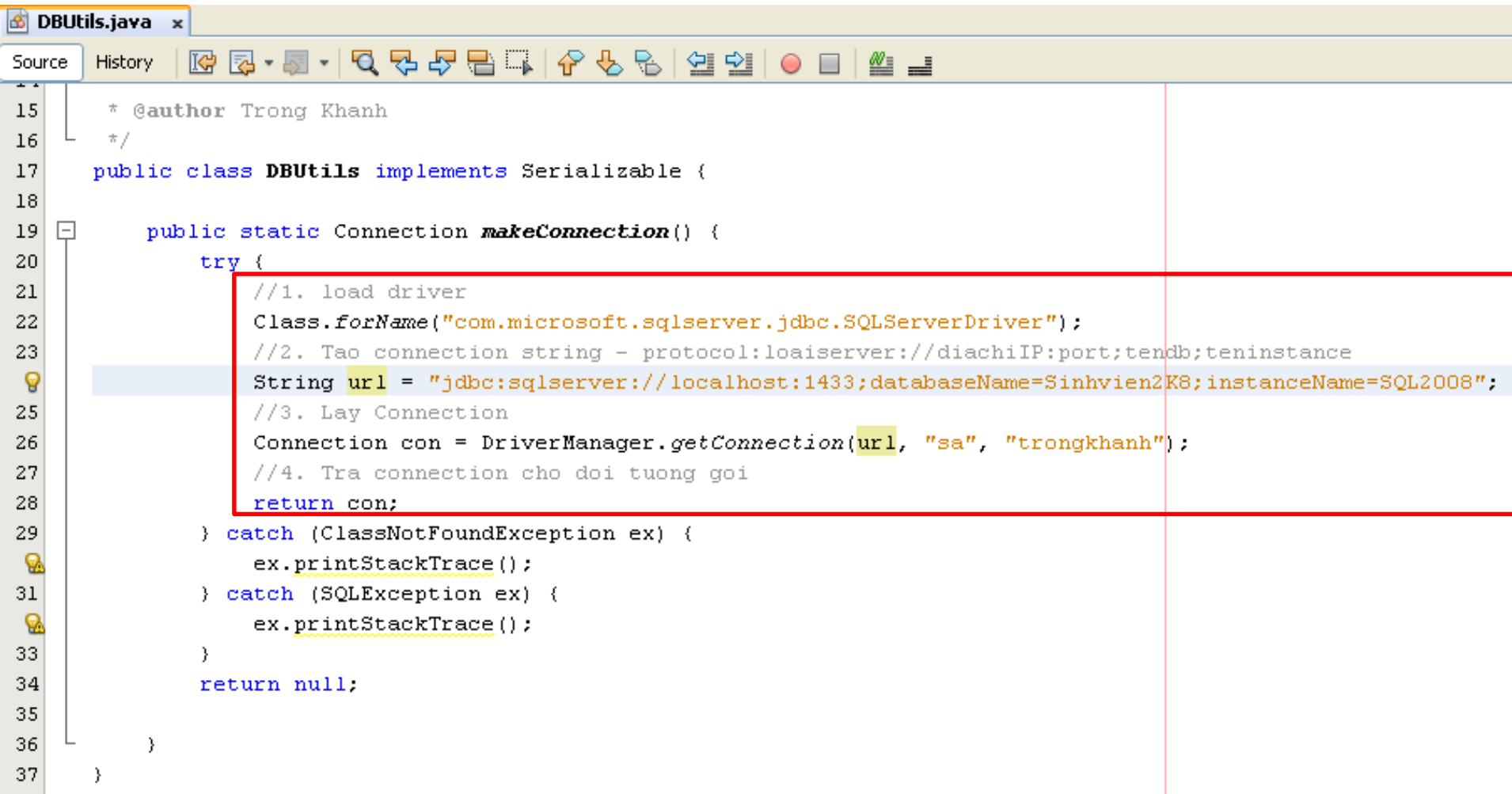
#JavaEE #MVC #MVC2

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



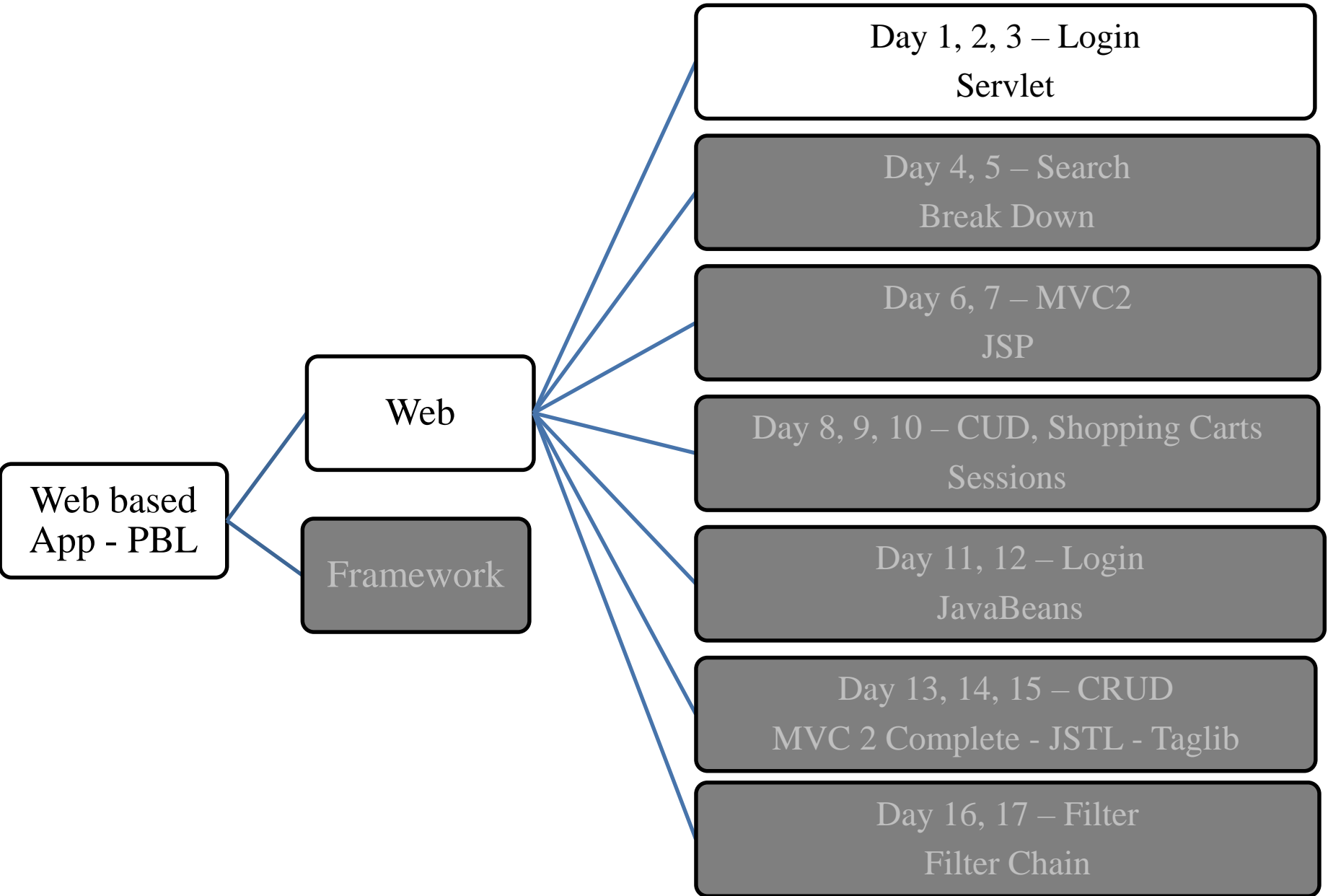
```

15  * @author Trong Khanh
16  */
17  public class DBUtils implements Serializable {
18
19      public static Connection makeConnection() {
20          try {
21              //1. load driver
22              Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
23              //2. Tao connection string - protocol:loaiserver://diachiIP:port;tendb;teninstance
24              String url = "jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K8;instanceName=SQL2008";
25              //3. Lay Connection
26              Connection con = DriverManager.getConnection(url, "sa", "trongkhanh");
27              //4. Tra connection cho doi tuong goi
28              return con;
29          } catch (ClassNotFoundException ex) {
30              ex.printStackTrace();
31          } catch (SQLException ex) {
32              ex.printStackTrace();
33          }
34          return null;
35      }
36  }
37  }
    
```

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

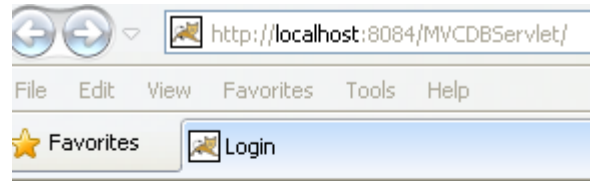
Objectives



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

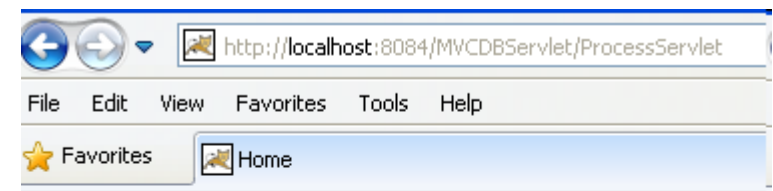
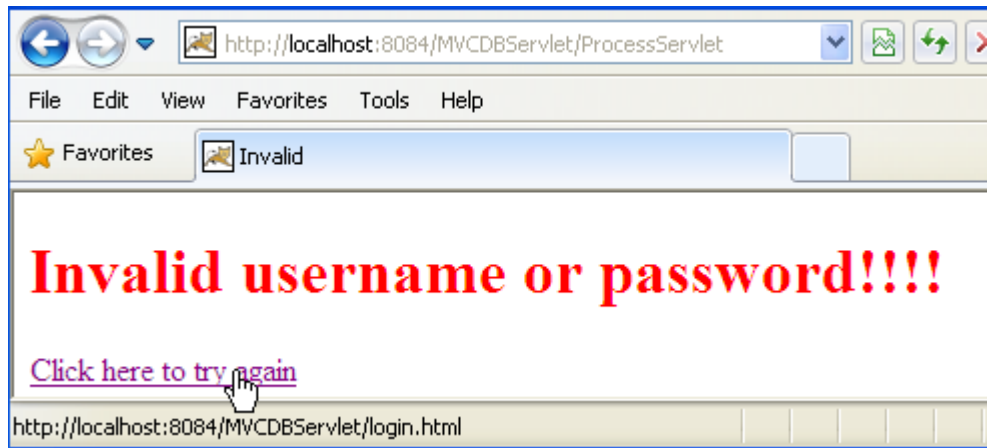
Build The Simple Web Expectation



Login Page

Username

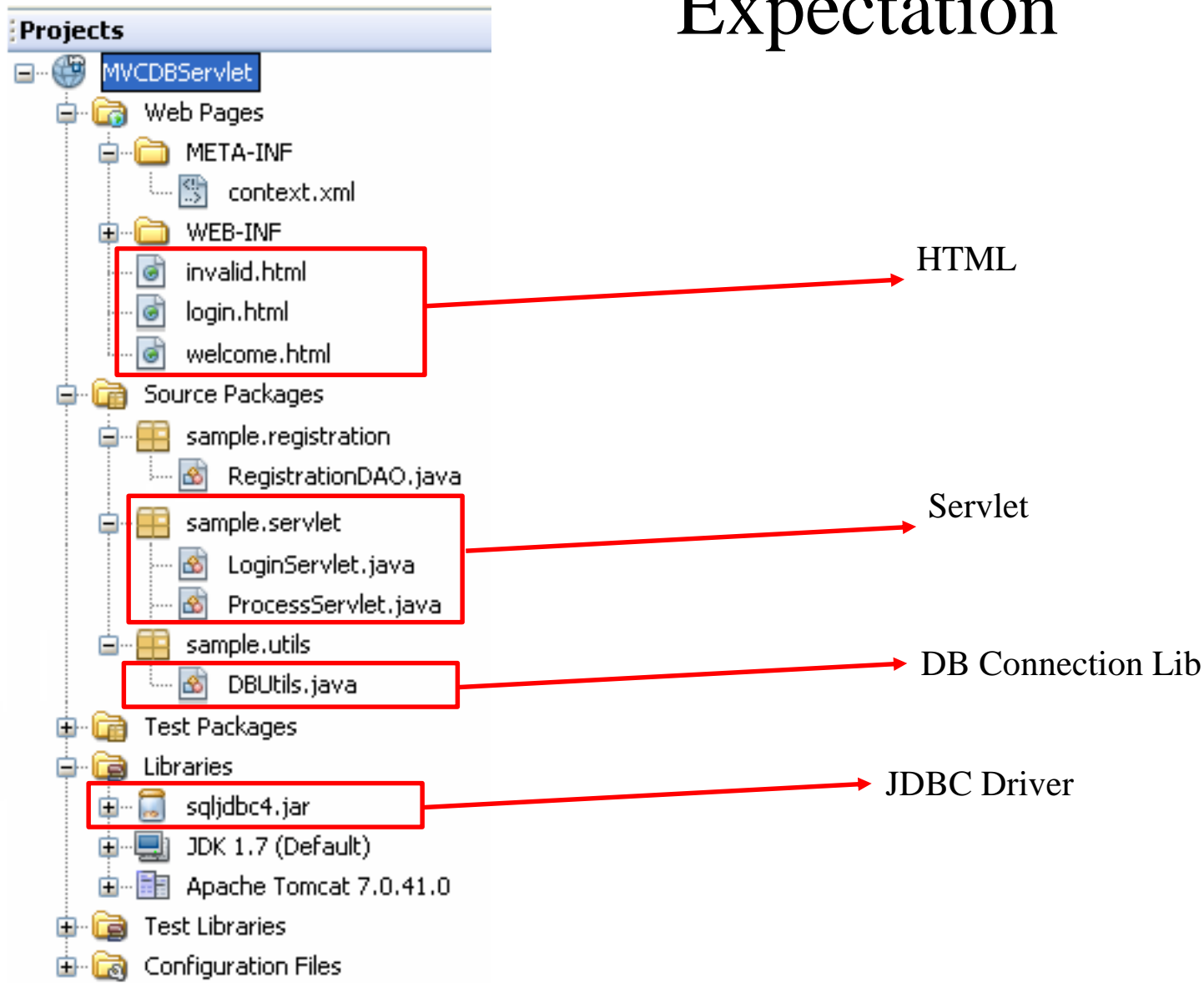
Password



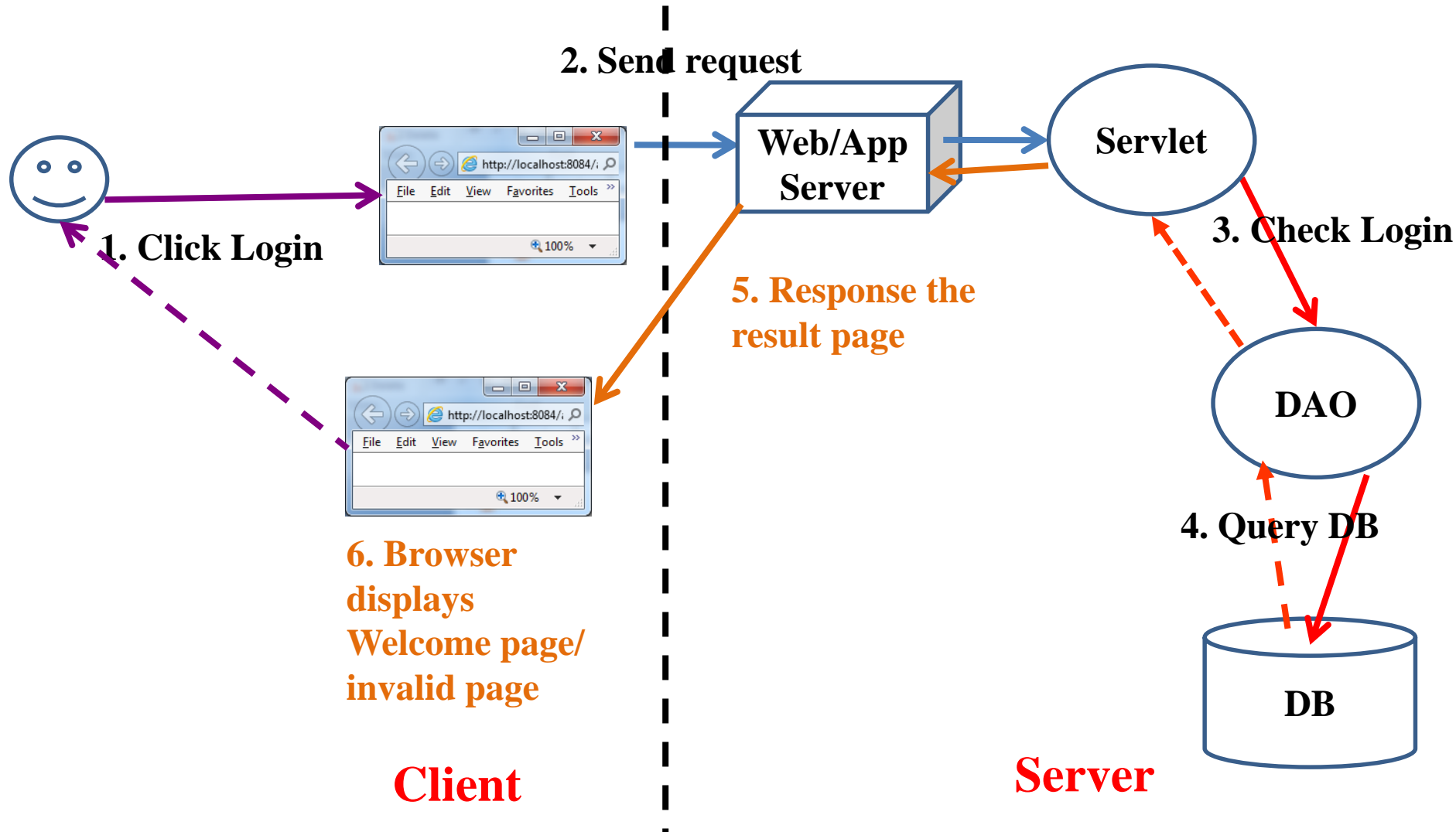
Welcome to DB Servlet

Name

Build The Simple Web Expectation

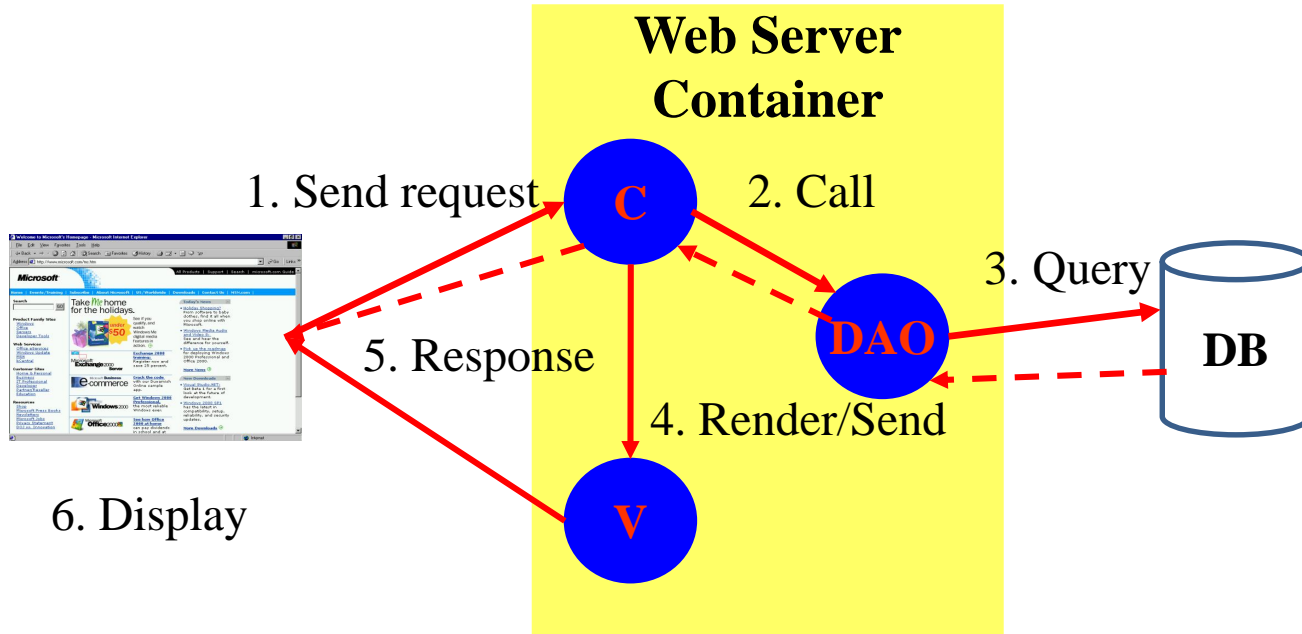


Build The Simple Web Interactive Server Model



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 **Hyper Text Markup Language**
 - 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 `` and ``
 - 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

```

1  <!--
2  To change this template, choose Tools | Templates
3  and open the template in the editor.
4  -->
5  <!DOCTYPE html>
6  <html>
7  <head>
8      <title>HTML</title>
9      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10 </head>
11 <body>
12     <h1>My first heading</h1>
13     <p>My first paragraph</p>
14     <a href="https://www.facebook.com/TrongKhanh.Kieu">My Website</a> <br/>
15     Or <a href="http://www.kieutrongkhanh.net">My Alternative Website</a>
16 </body>
17 </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 **assume** 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
 - Dedicated 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 sort 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

```

formParameters.html x
Source History
11 <body>
12     <h1>HTML Forms</h1>
13     <form action="index.html">
14         Textbox <input type="text" name="txtText" value="" size="5" /><br/>
15         Password <input type="password" name="txtPassword" value="" /><br/>
16         Hidden <input type="hidden" name="txtHidden" value="" /><br/>
17         Male <input type="checkbox" name="chkCheck" value="ON" checked="checked" /><br/>
18         Status
19         <input type="radio" name="rdoStatus" value="Single" checked="checked" />Single<br/>
20         <input type="radio" name="rdoStatus" value="Married" />Married<br/>
21         <input type="radio" name="rdoStatus" value="Divorced" />Divorced<br/>
22         ComboBox <select name="txtCombo">
23             <option value="Servlet">JSP and Servlet</option>
24             <option value="EJB">EJB</option>
25         </select><br/>
26         Multiple <select name="txtList" multiple="multiple" size="3">
27             <option value="Servlet" selected>JSP and Servlet</option>
28             <option value="EJB" selected>EJB</option>
29             <option value="Java">Core Java</option>
30         </select><br/>
31         TextArea <textarea name="txtArea" rows="4" cols="20">
32             This is a form parameters demo!!!!
33         </textarea><br/>
34         <input type="submit" name="txtB" />
35         <input type="submit" value="Register" name="action" />
36         <input type="reset" name="txtB" />
37         <input type="button" value="JavaScript" name="txtB" onclick="" />
38     </form>
39 </body>
40 </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

☐ Divorced

ComboBox JSP and Servlet ▼

JSP and Servlet

EJB

Multiple Core Java

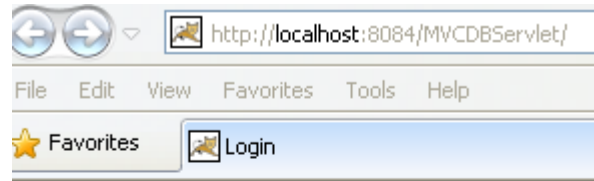
TextArea

This is a form parameters demo!!!!

Submit Query Register Reset JavaScript

Done

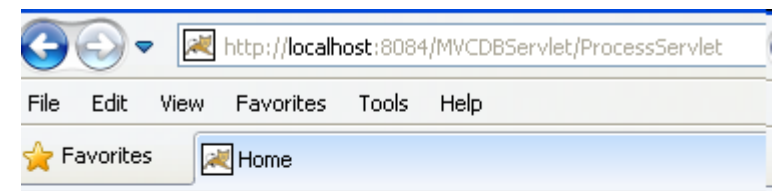
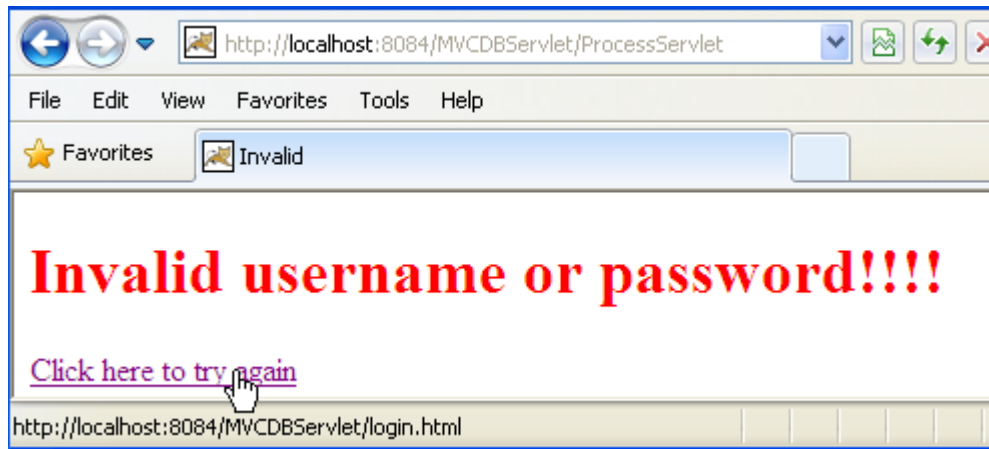
Build The Simple Web Views



Login Page

Username

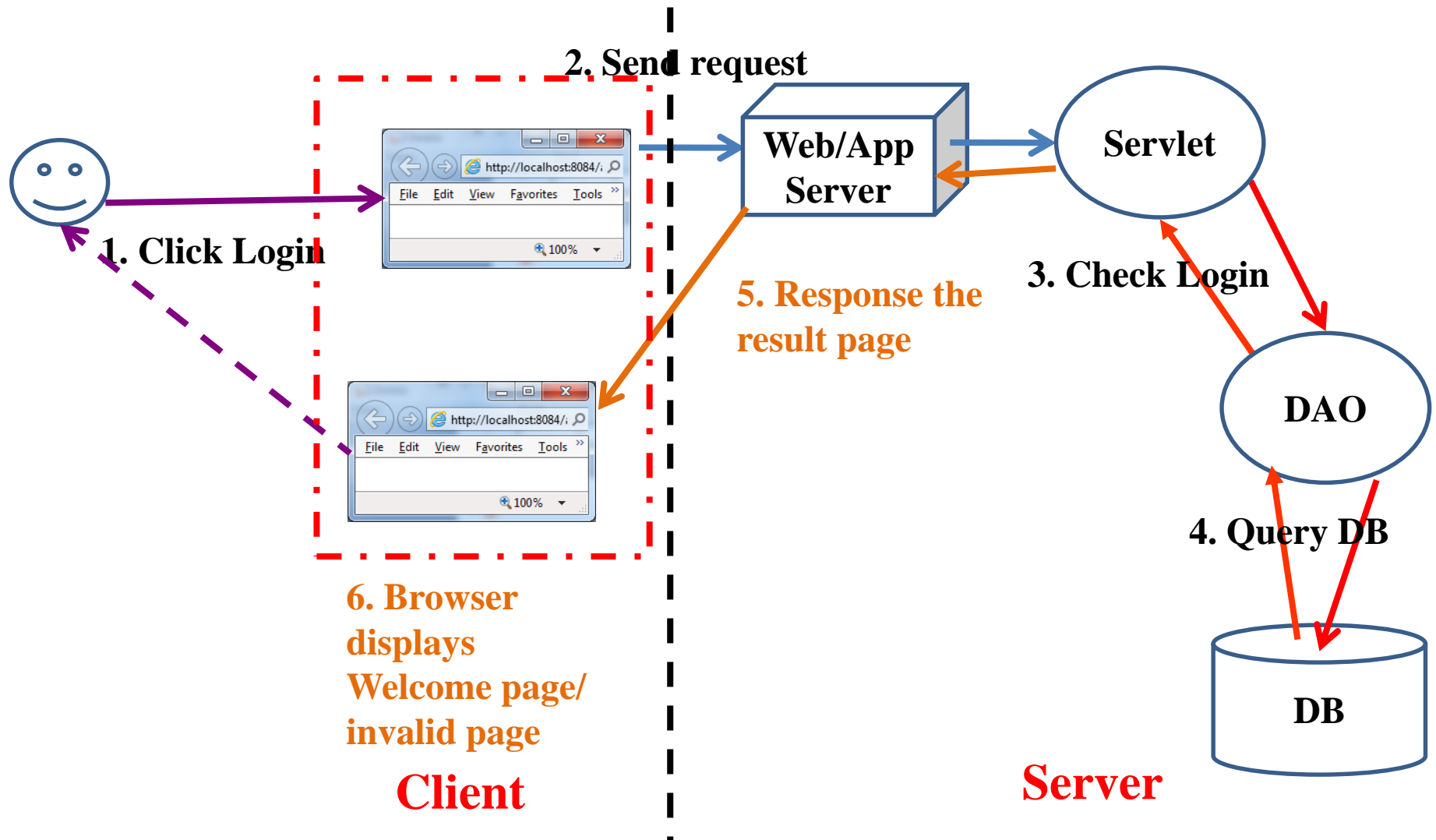
Password



Welcome to DB Servlet

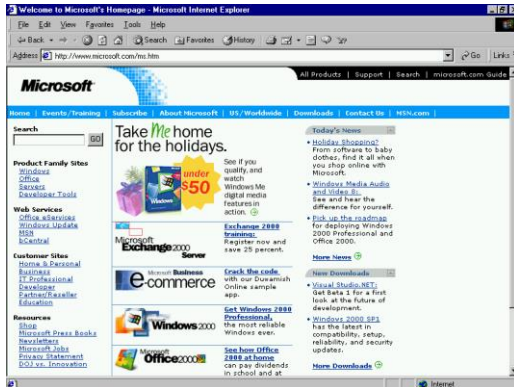
Name

Build The Simple Web Interactive Server Model



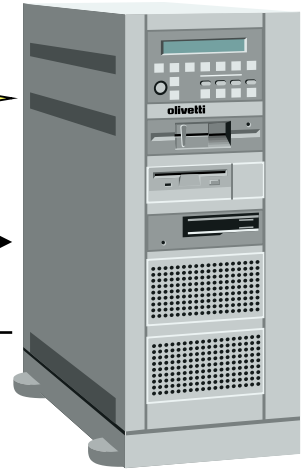
HTTP Protocols Overview

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



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

4. The result is **responded** to Browser



<http://microsoft.com/index.html>

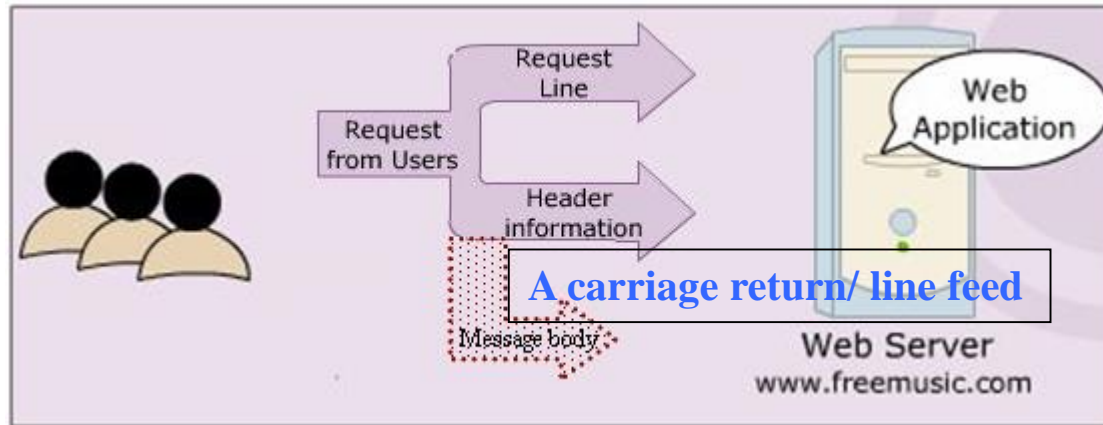
5. Web Browser views the result which contains a markup language

3. **192.168.54.3:80** Web Server processes a request (connecting DB, calculating, call service ...)

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

HTTP Protocols

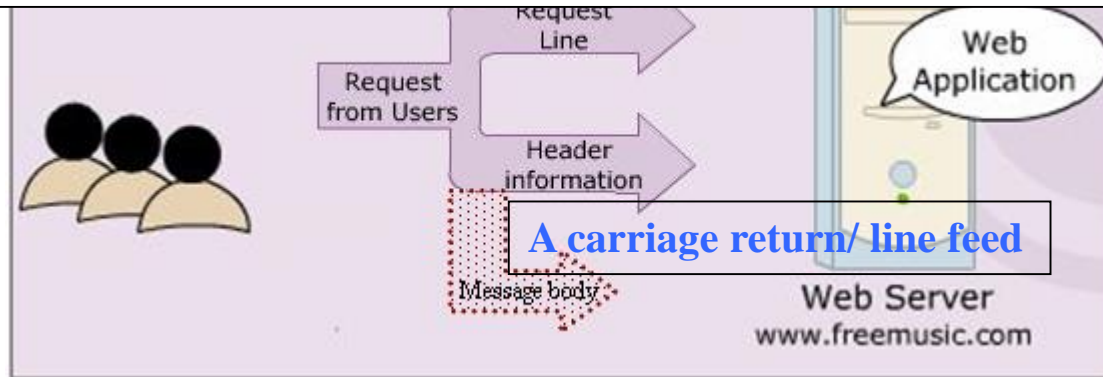
HTTP Requests



HTTP Protocols

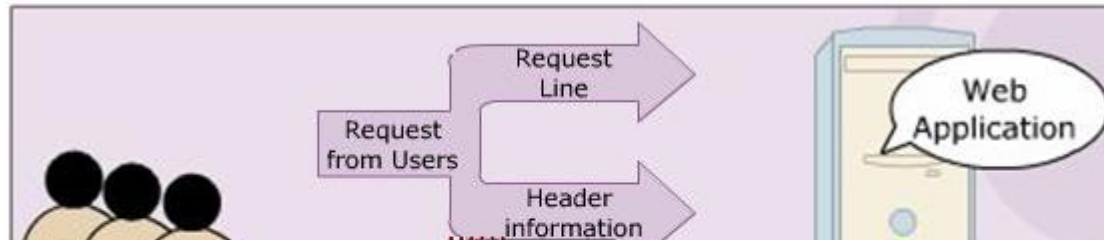
HTTP Requests

- **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**



HTTP Protocols

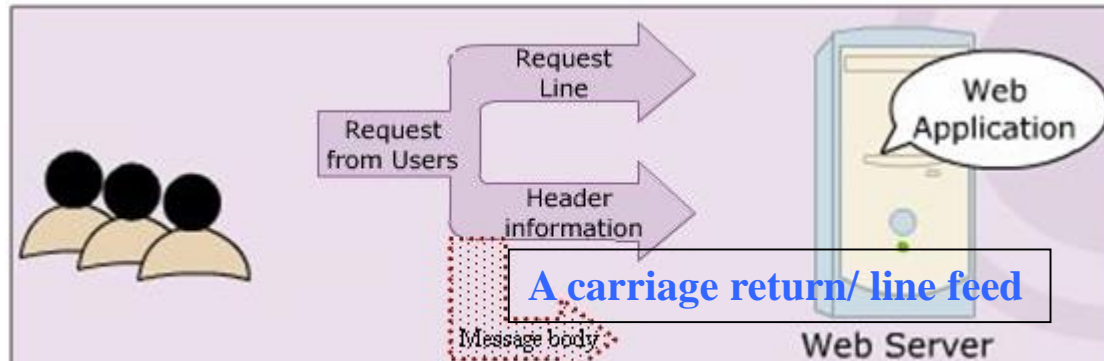
HTTP Requests



- 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



- 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=2A307CB619854E2FOODDF9630BE91DA7

HTTP Protocols

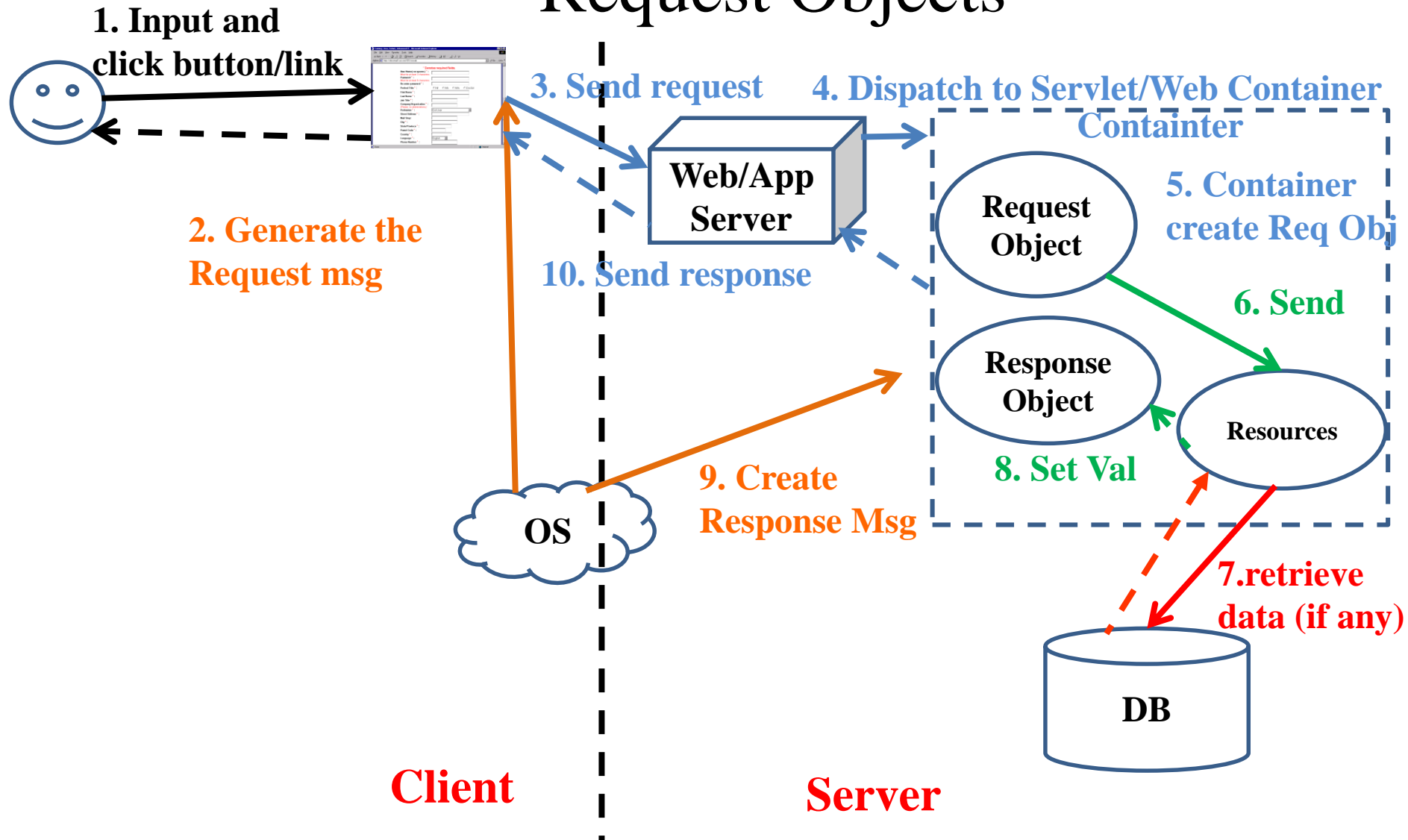
HTTP Requests – Example

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
```

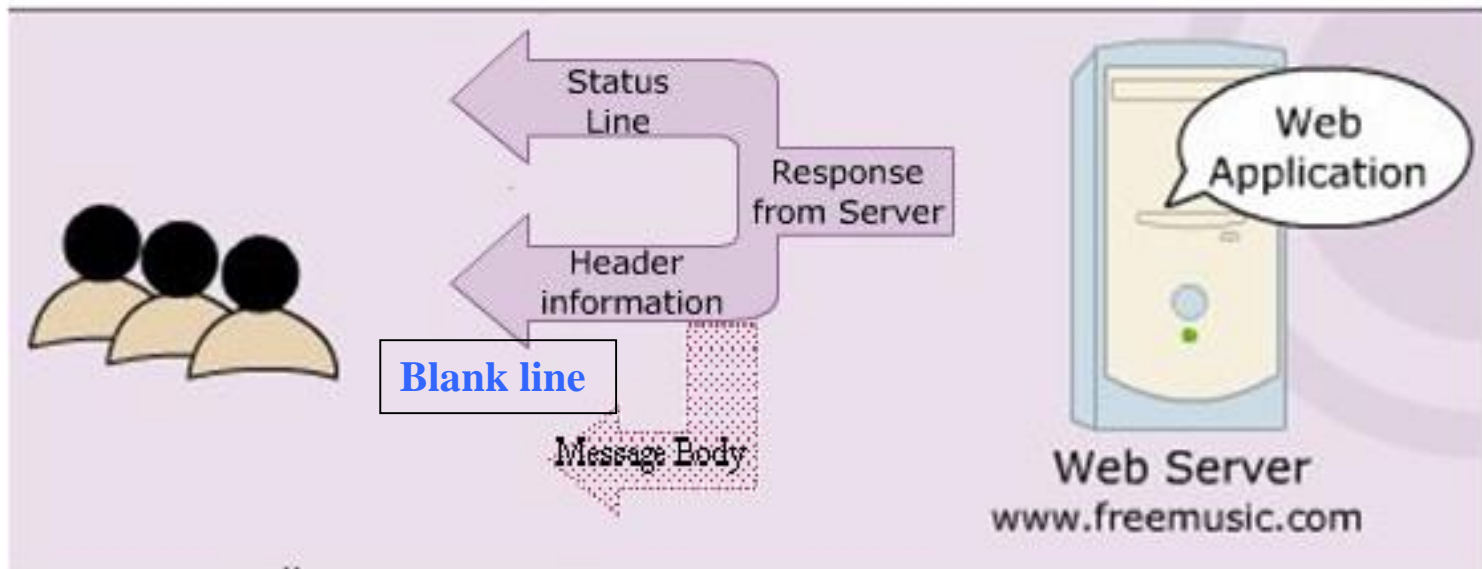
The Servlet Model

Request Objects



HTTP Protocols

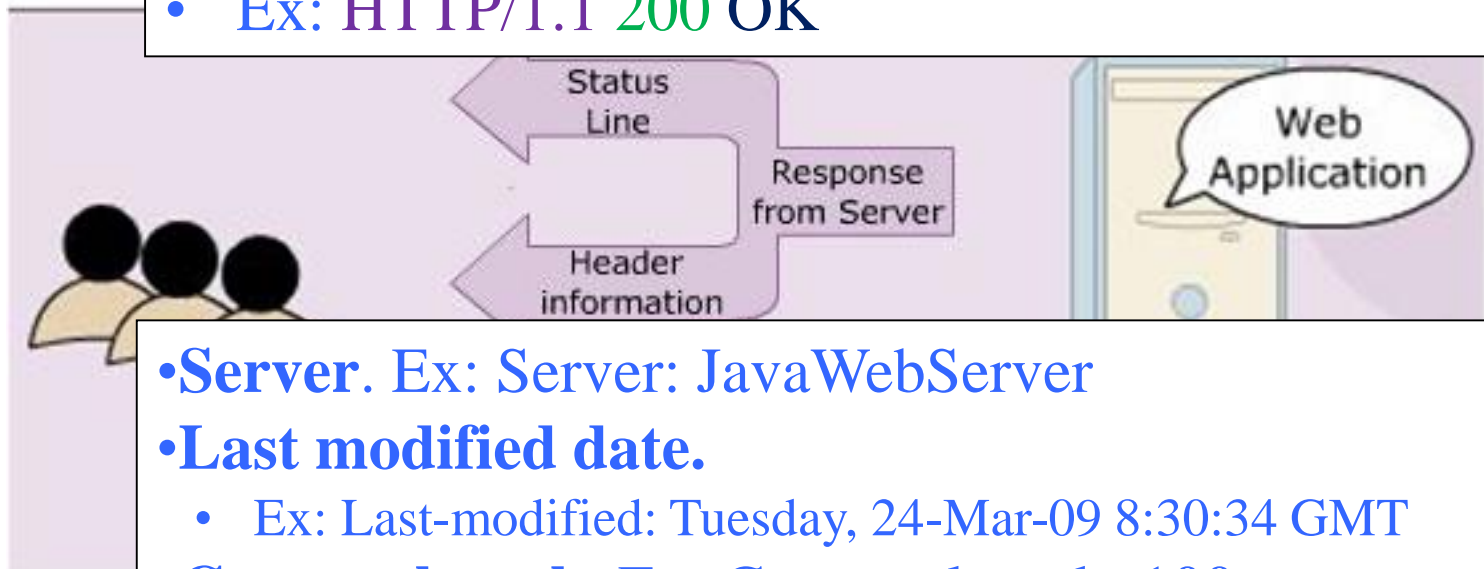
HTTP Responses



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	Moved Permanently	- 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

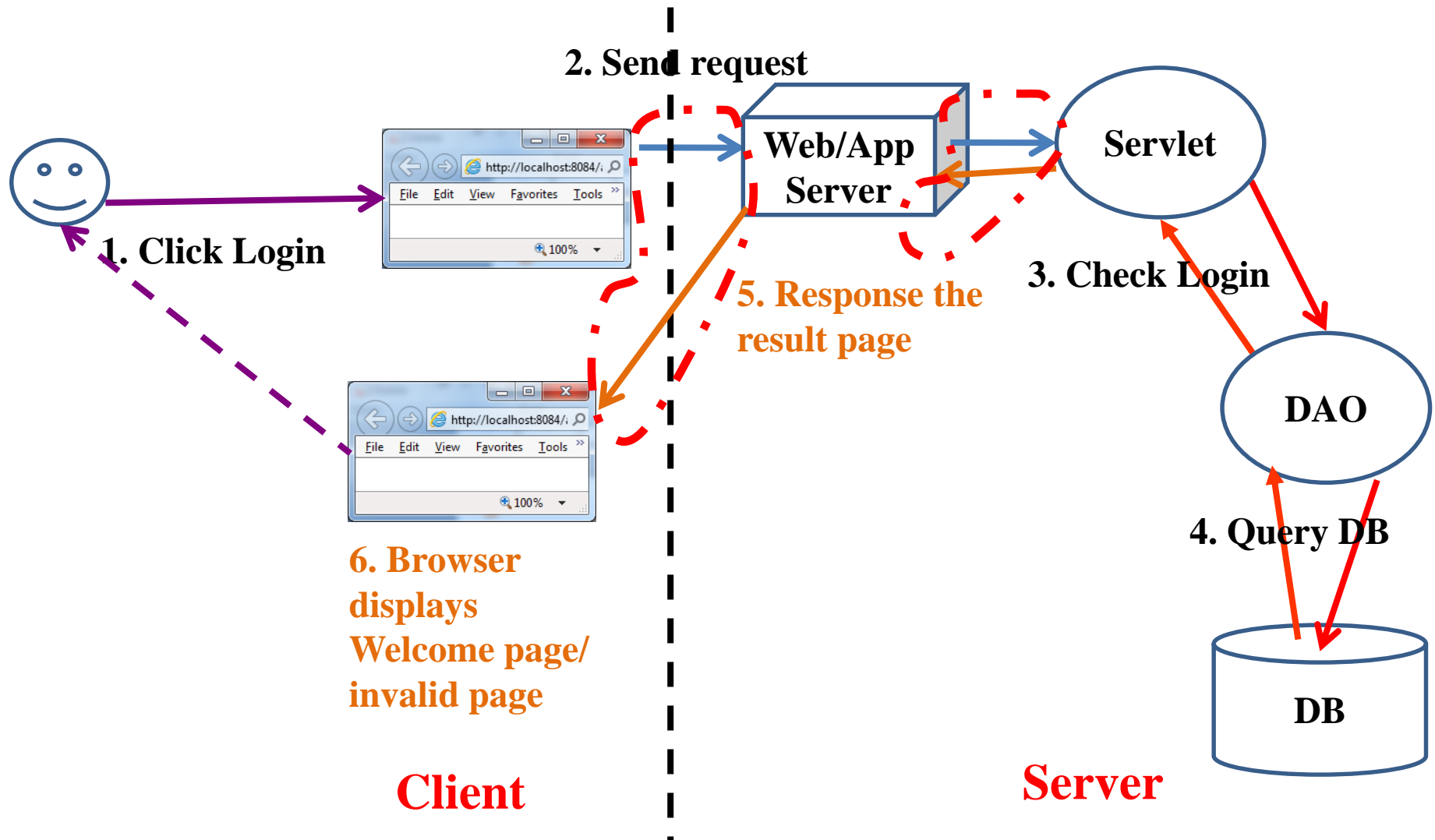
HTTP Protocols

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

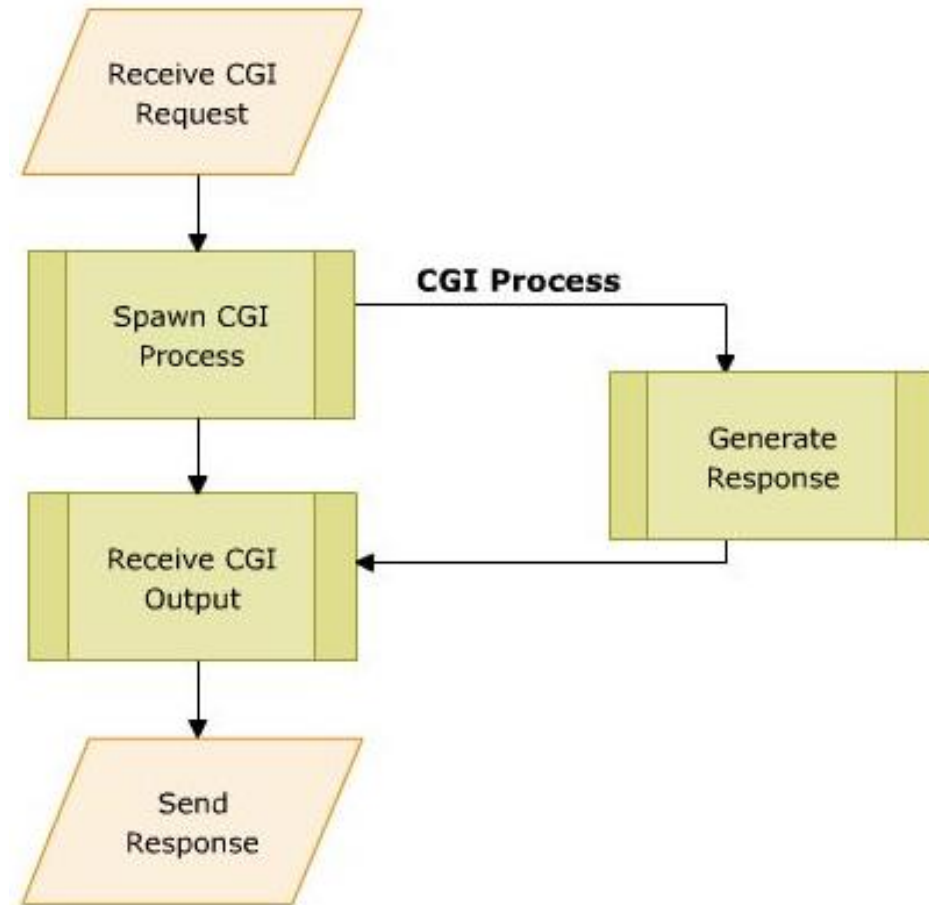
Interactive Server Model



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**.

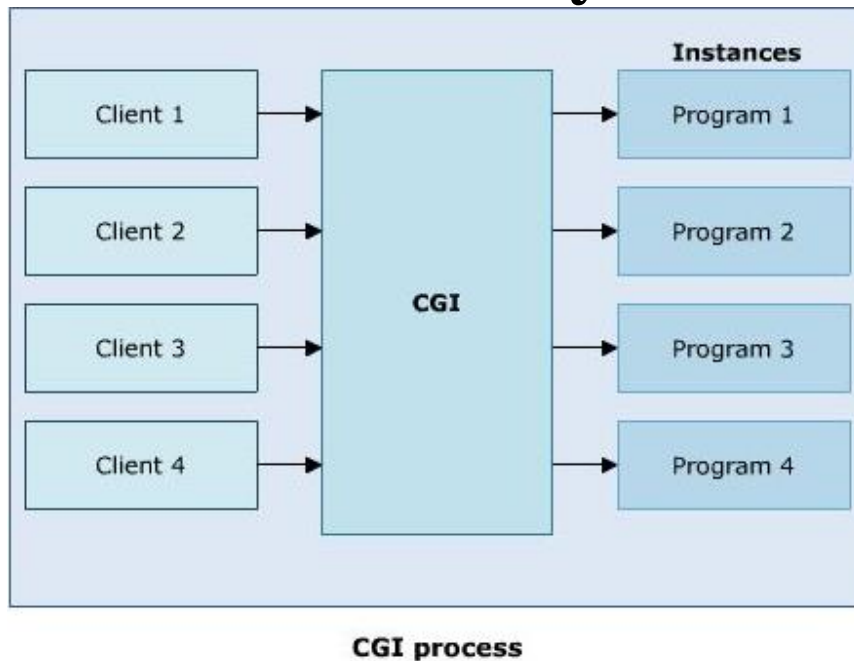


Server Process for running CGI

The Servlet Model

Common Gateway Interface (CGI)

- Disadvantages
 - **Reduced efficiency**



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 a request, 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>
```

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

```
  <servlet-name>servlet name</servlet-name>
```

```
  <servlet-class>package.classname</servlet-class>
```

```
</servlet>
```

```
<servlet-mapping>
```

Define the access path to the servlet;

```
  <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>
```

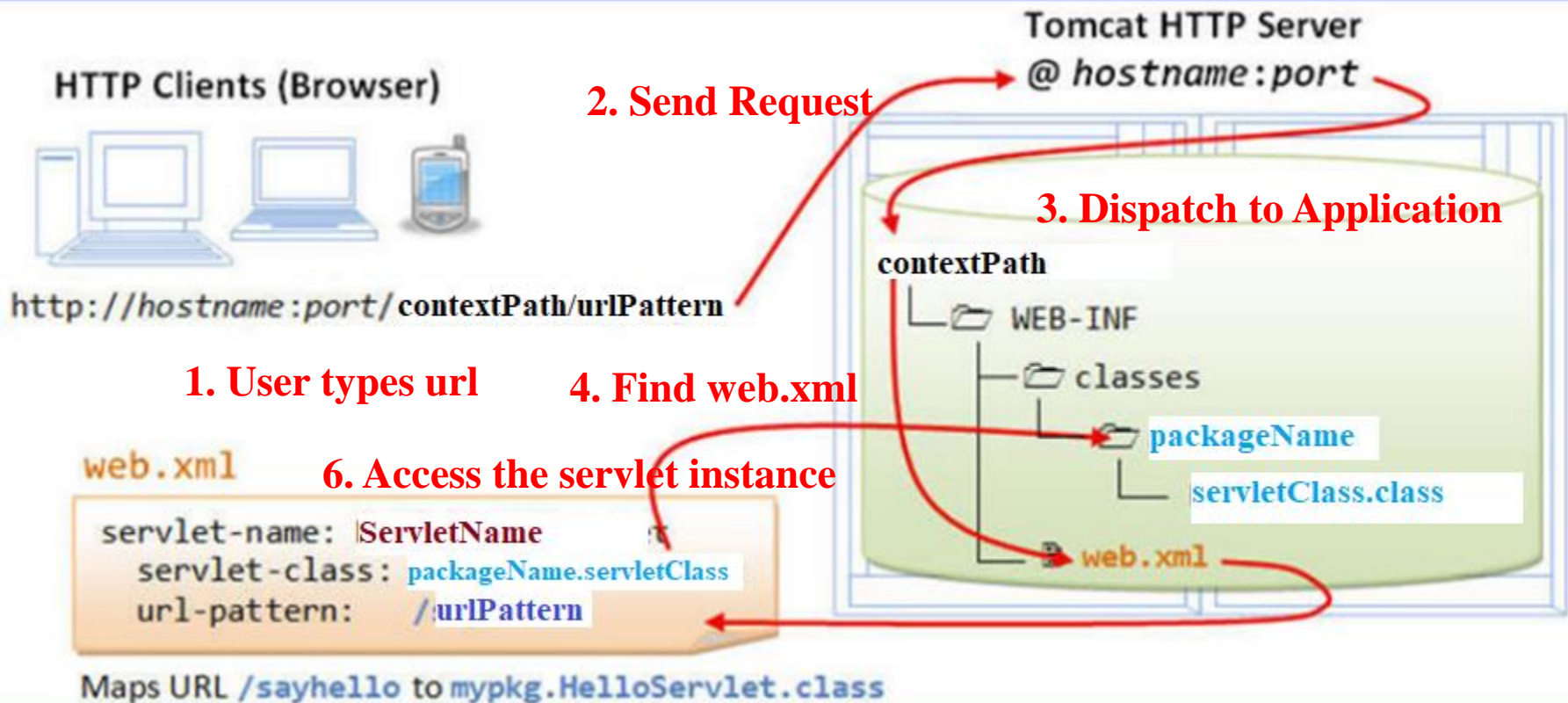
The Servlet Model

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



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

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

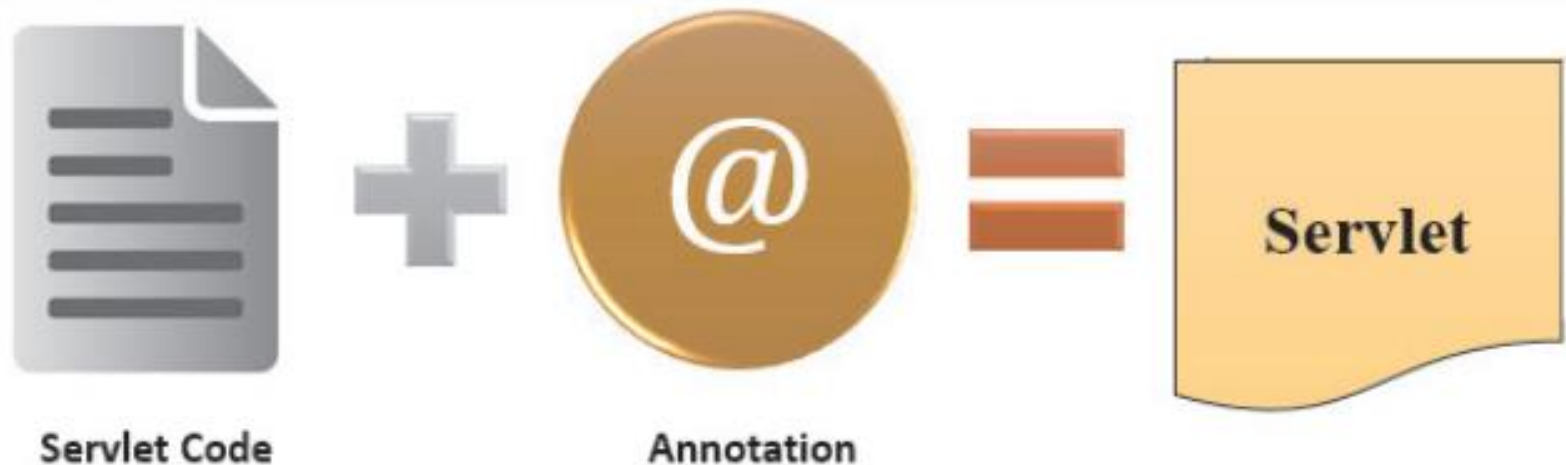


Figure 4.9: Servlet with Annotations, Web Component Development Using Java, Aptech World Wide

The Servlet Model

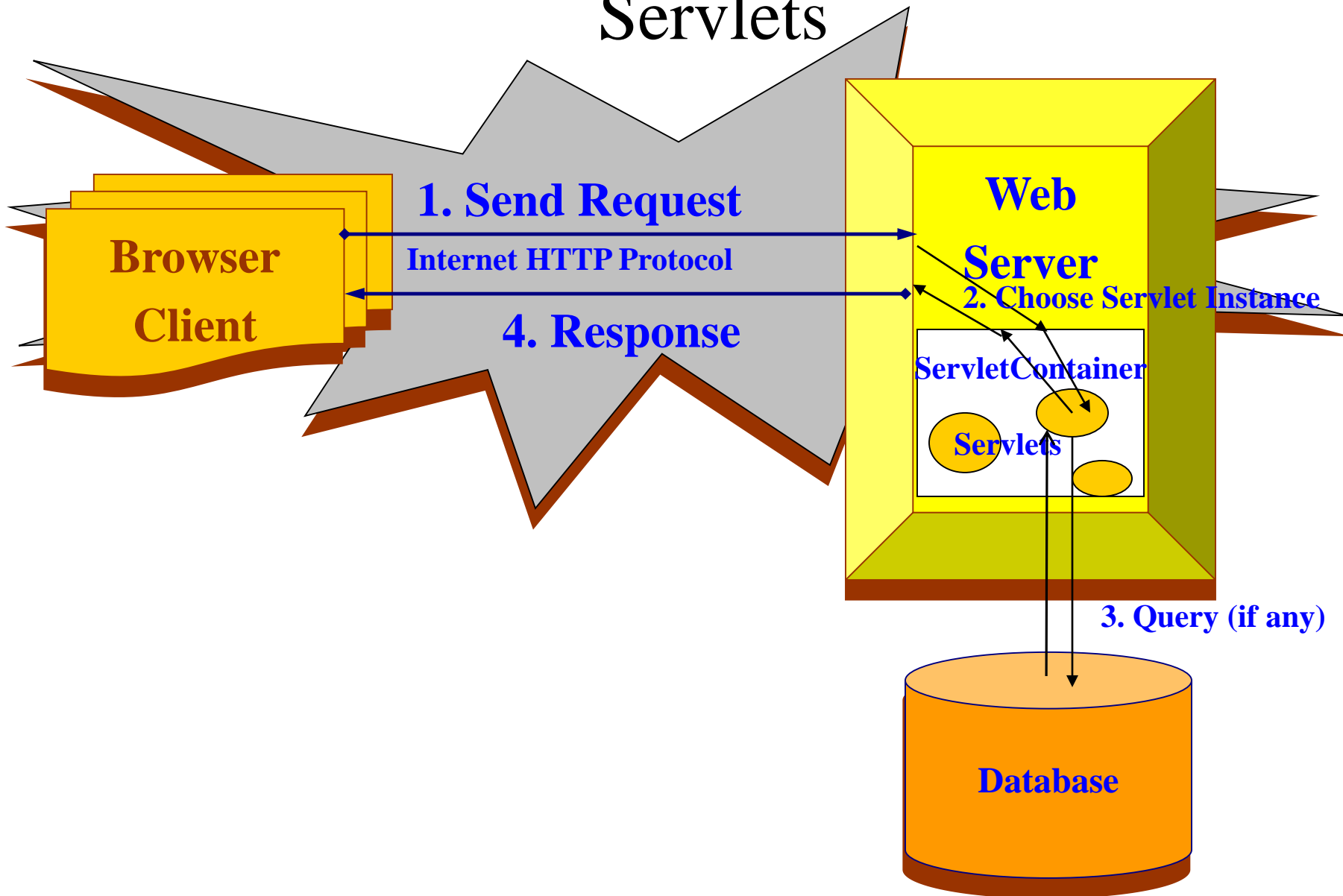
Annotations – Servlets

- **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

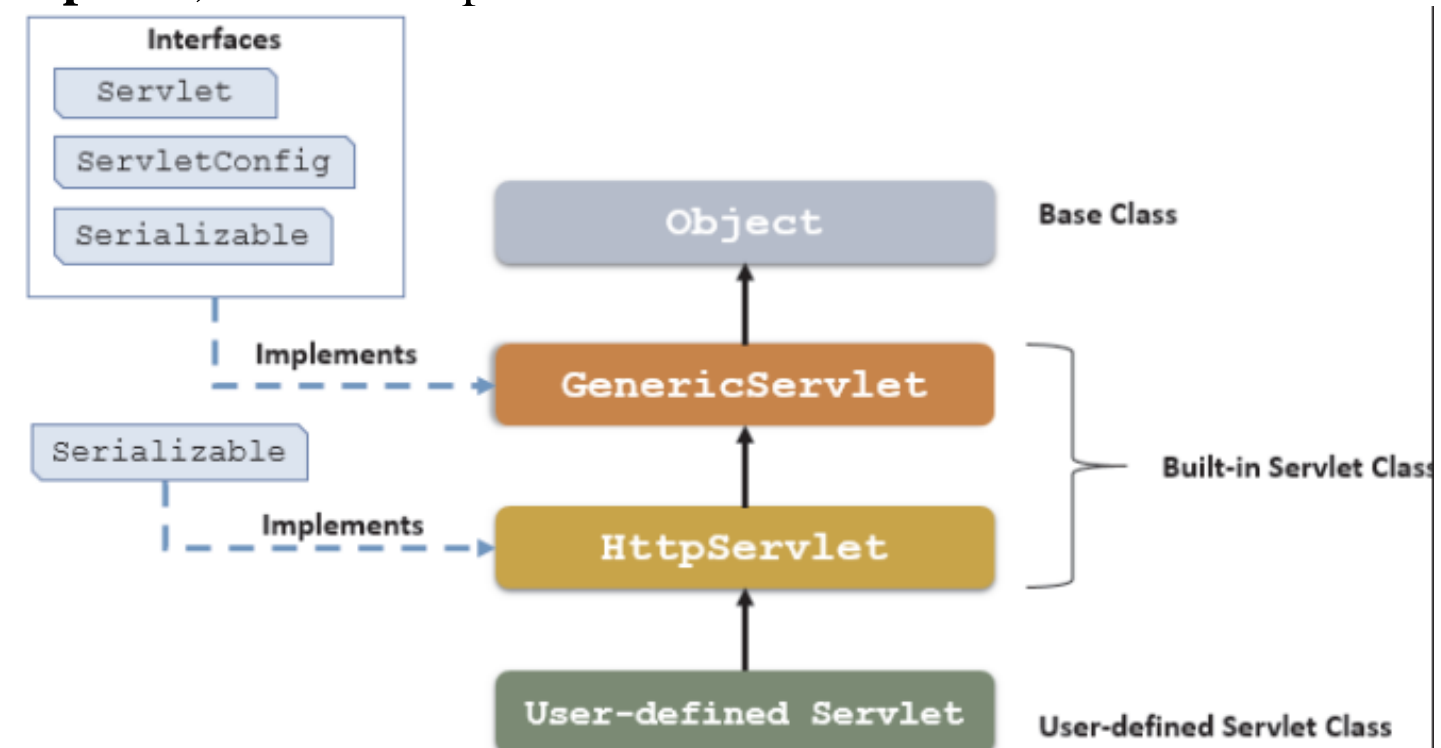
Servlets



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

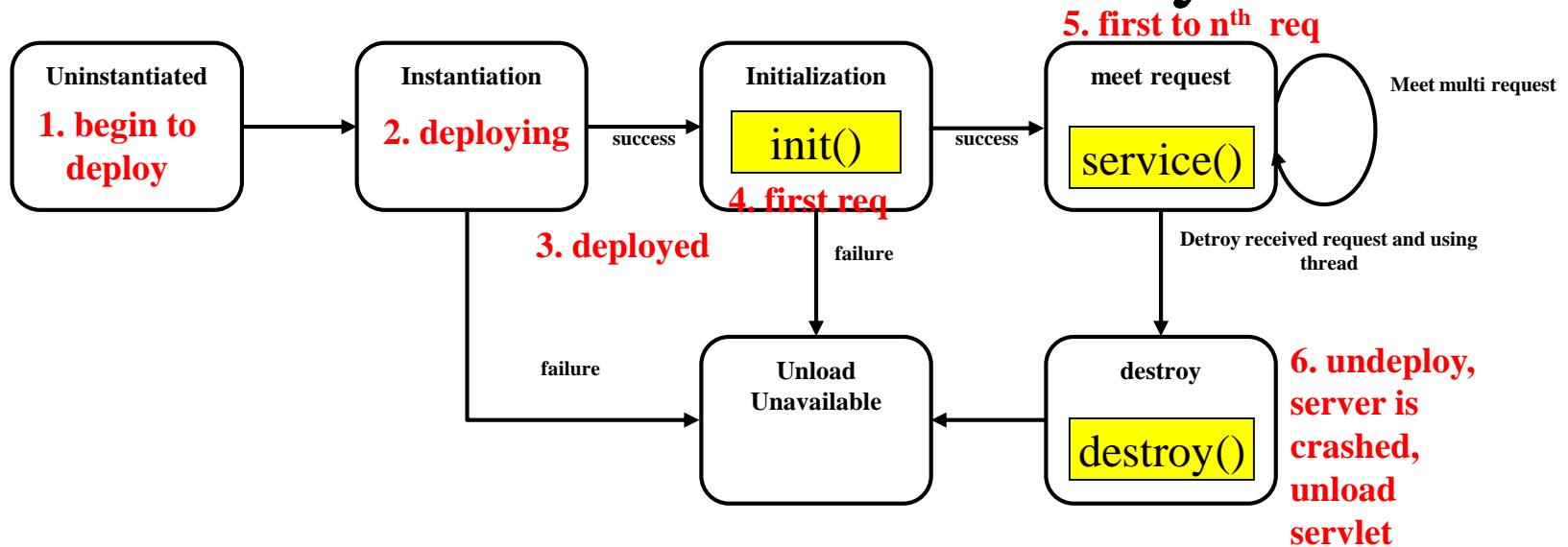
GenericServlet class

- 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	<ul style="list-style-type: none"> - public void init() throws ServletException - Initializes the servlet
service <div>Servlet Life Cycle defined in GenericServlet</div>	<ul style="list-style-type: none"> - public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, IOException - Called by the container to respond to a servlet request
destroy	<ul style="list-style-type: none"> - public void destroy(): cleaning the servlet

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

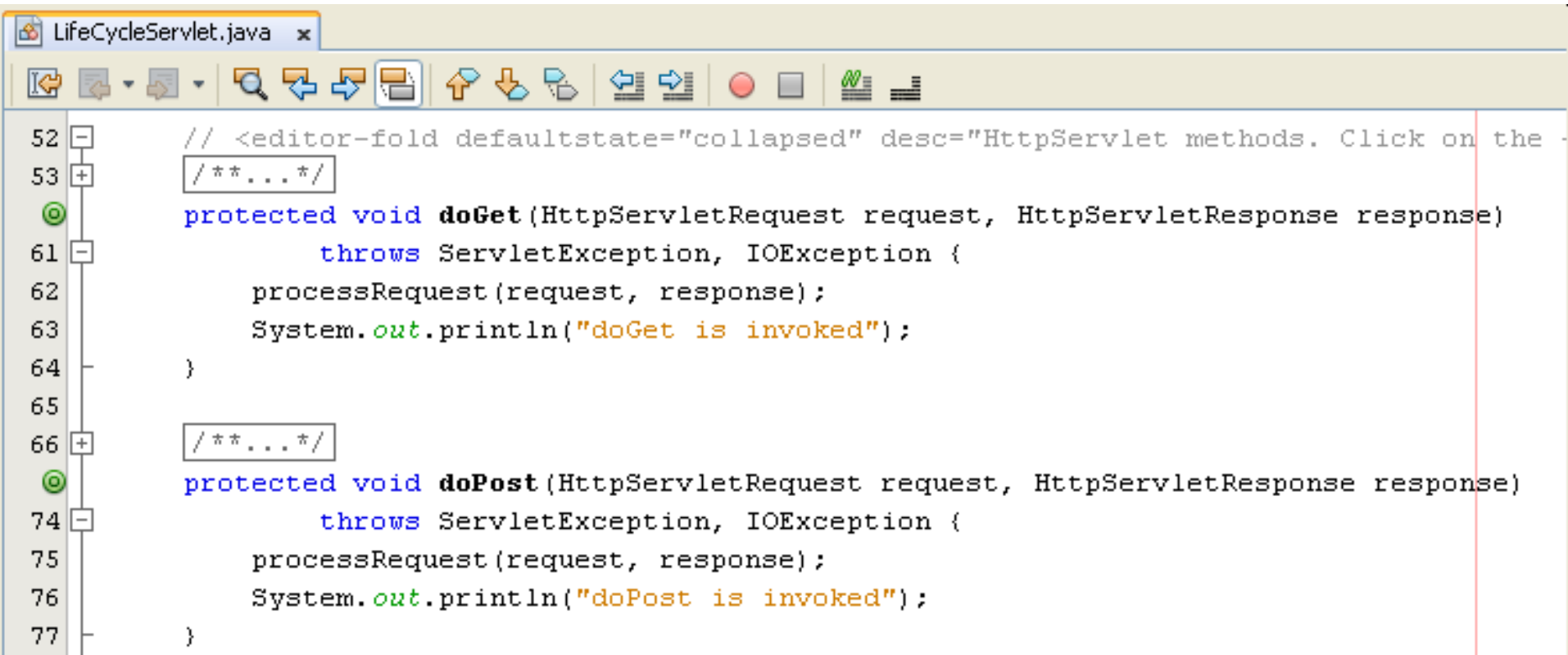
The Servlet Life Cycle – Example

```

LifeCycleServlet.java x
16  * @author Trong Khanh
17  */
18  public class LifeCycleServlet extends HttpServlet {
19      private int a = 0;
20      public void init() throws ServletException {
21          super.init();
22          System.out.println("init");
23          a += 5;
24          System.out.println("a = " + a);
25      }
26      /**...*/
33      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
34          throws ServletException, IOException {
35          response.setContentType("text/html;charset=UTF-8");
36          PrintWriter out = response.getWriter();
37          try {
38              out.println("<html>");
39              out.println("<head>");
40              out.println("<title>Servlet</title>");
41              out.println("</head>");
42              out.println("<body>");
43              out.println("<h1>Servlet Life Cycle</h1>");
44
45              a += 10;
46              out.println("a = " + a);
47          } finally {
48              out.close();
49          }
50      }
  
```

The Servlet Model

The Servlet Life Cycle – Example



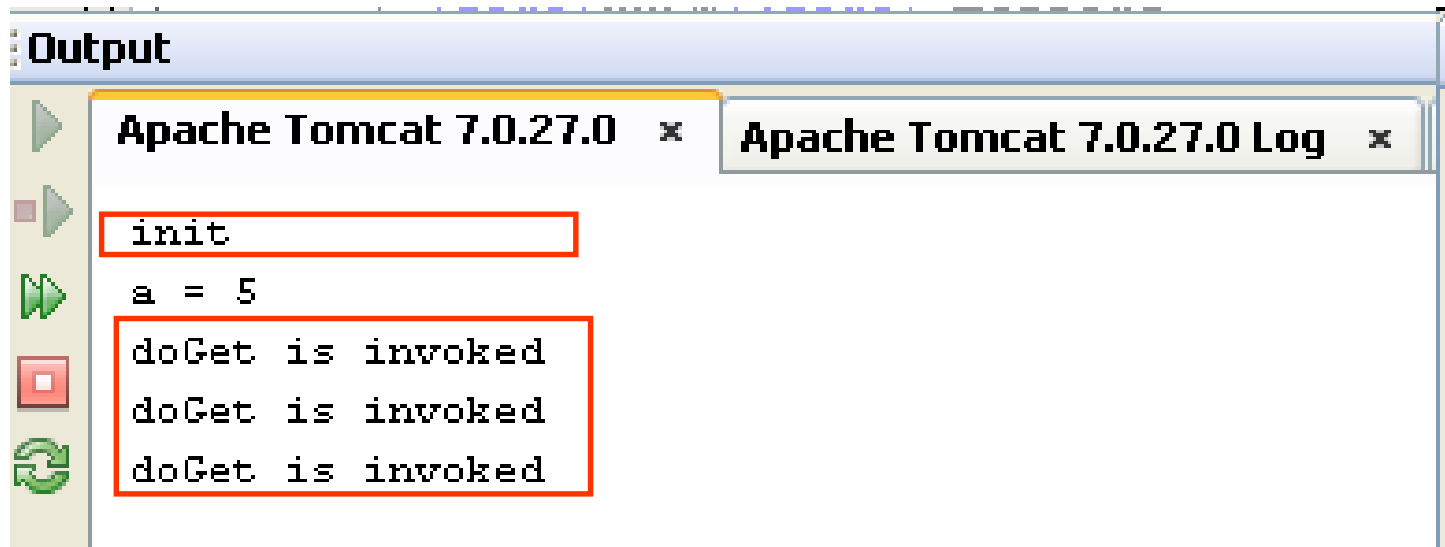
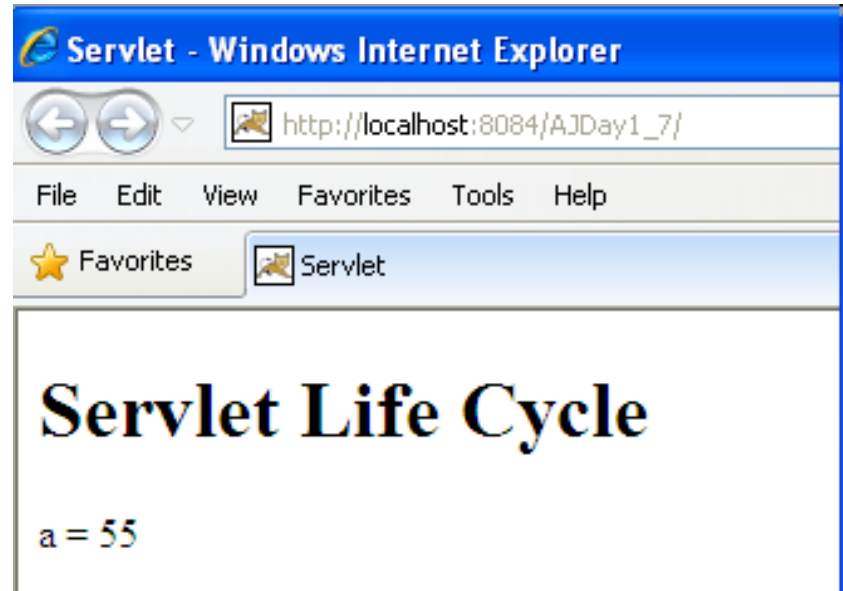
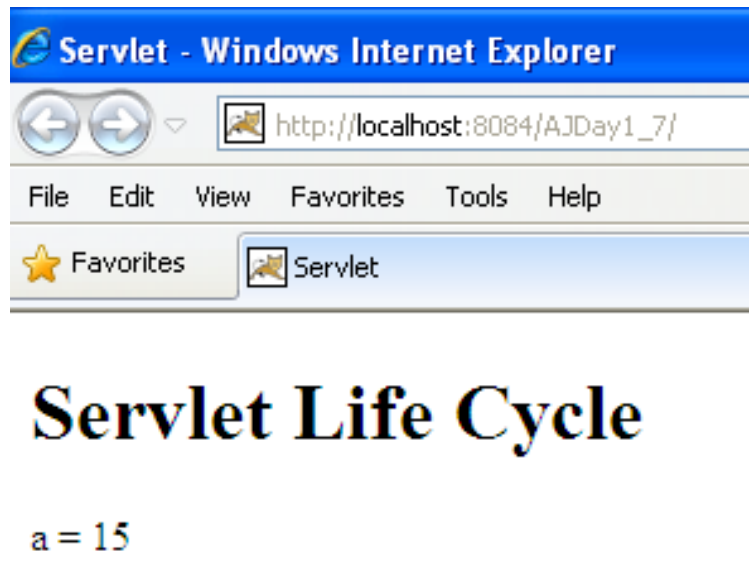
```

52 // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the
53 /**...*/
54 protected void doGet(HttpServletRequest request, HttpServletResponse response)
55     throws ServletException, IOException {
56     processRequest(request, response);
57     System.out.println("doGet is invoked");
58 }
59
60 /**...*/
61 protected void doPost(HttpServletRequest request, HttpServletResponse response)
62     throws ServletException, IOException {
63     processRequest(request, response);
64     System.out.println("doPost is invoked");
65 }
66

```

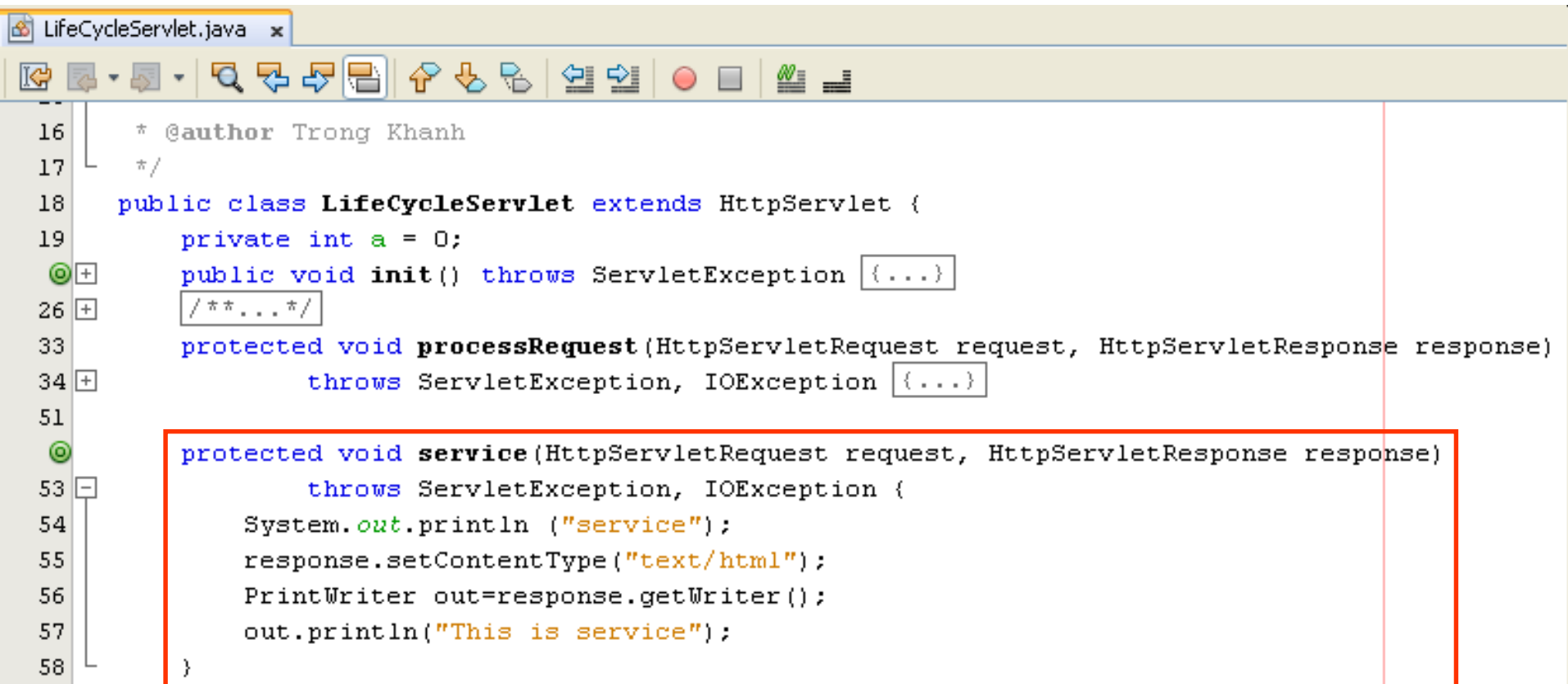

The Servlet Model

The Servlet Life Cycle – Example



The Servlet Model

The Servlet Life Cycle – Example

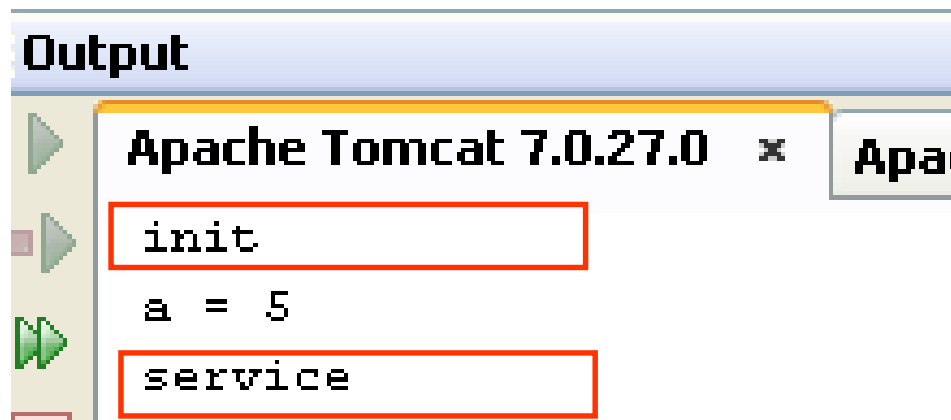
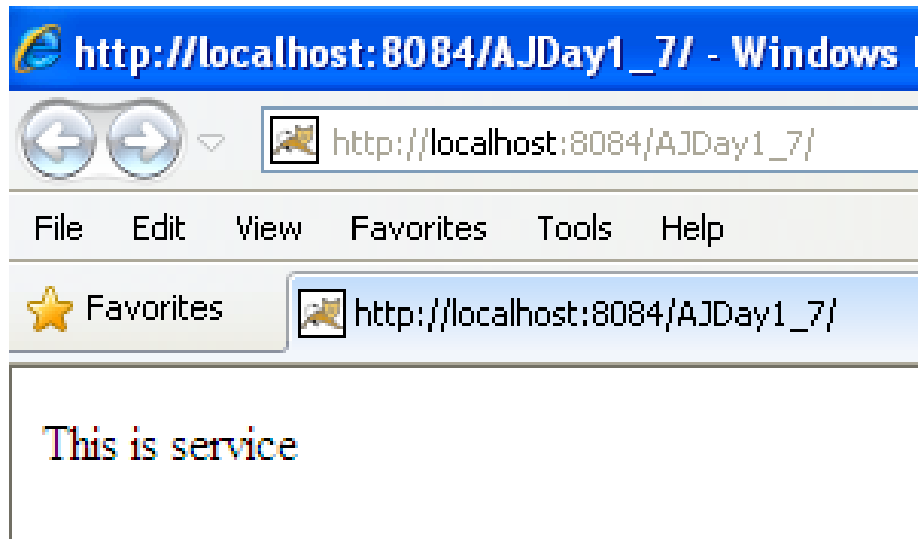


```

16  * @author Trong Khanh
17  */
18  public class LifeCycleServlet extends HttpServlet {
19      private int a = 0;
20      public void init() throws ServletException { ... }
21      /**...*/
22      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
23          throws ServletException, IOException { ... }
24
25      protected void service(HttpServletRequest request, HttpServletResponse response)
26          throws ServletException, IOException {
27          System.out.println ("service");
28          response.setContentType("text/html");
29          PrintWriter out=response.getWriter();
30          out.println("This is service");
31      }
32  }
    
```

The Servlet Model

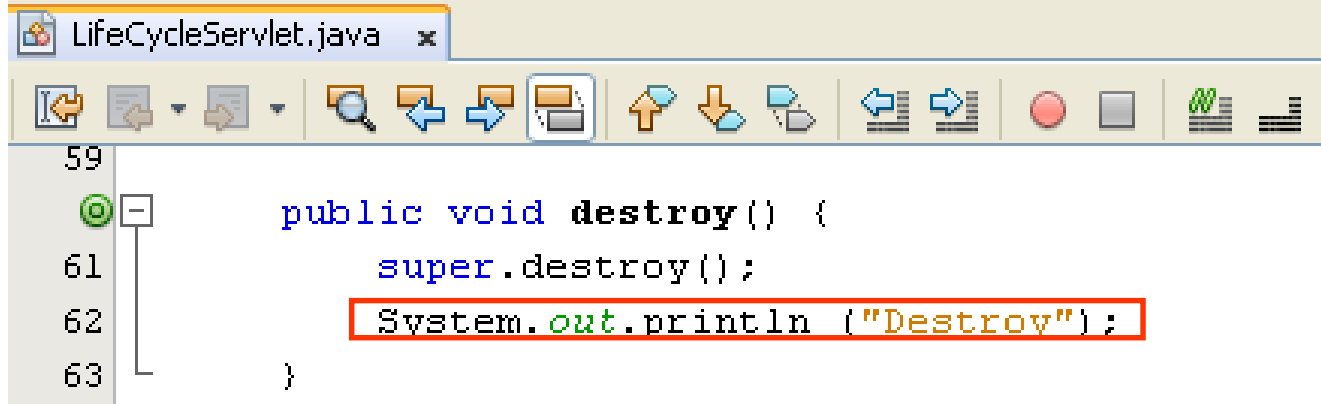
The Servlet Life Cycle – Example



The Servlet Model

The Servlet Life Cycle – Example

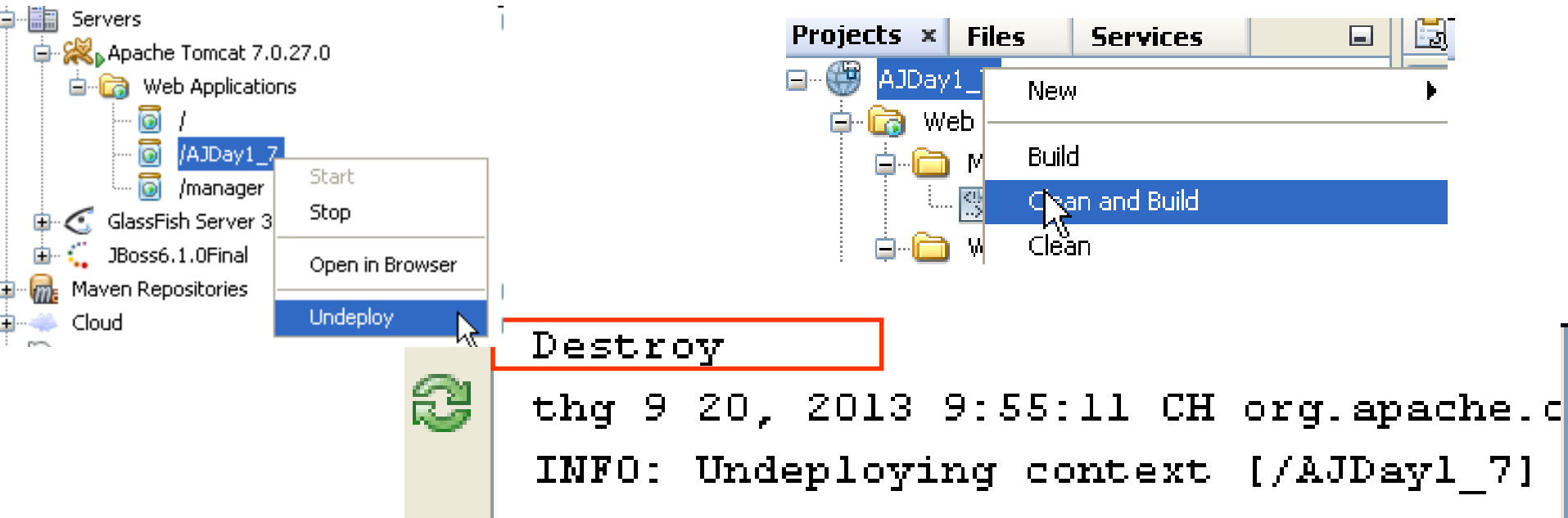
- Addition the destroy method (comment service method)



```

59
60 public void destroy() {
61     super.destroy();
62     System.out.println ("Destroy");
63 }
  
```

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



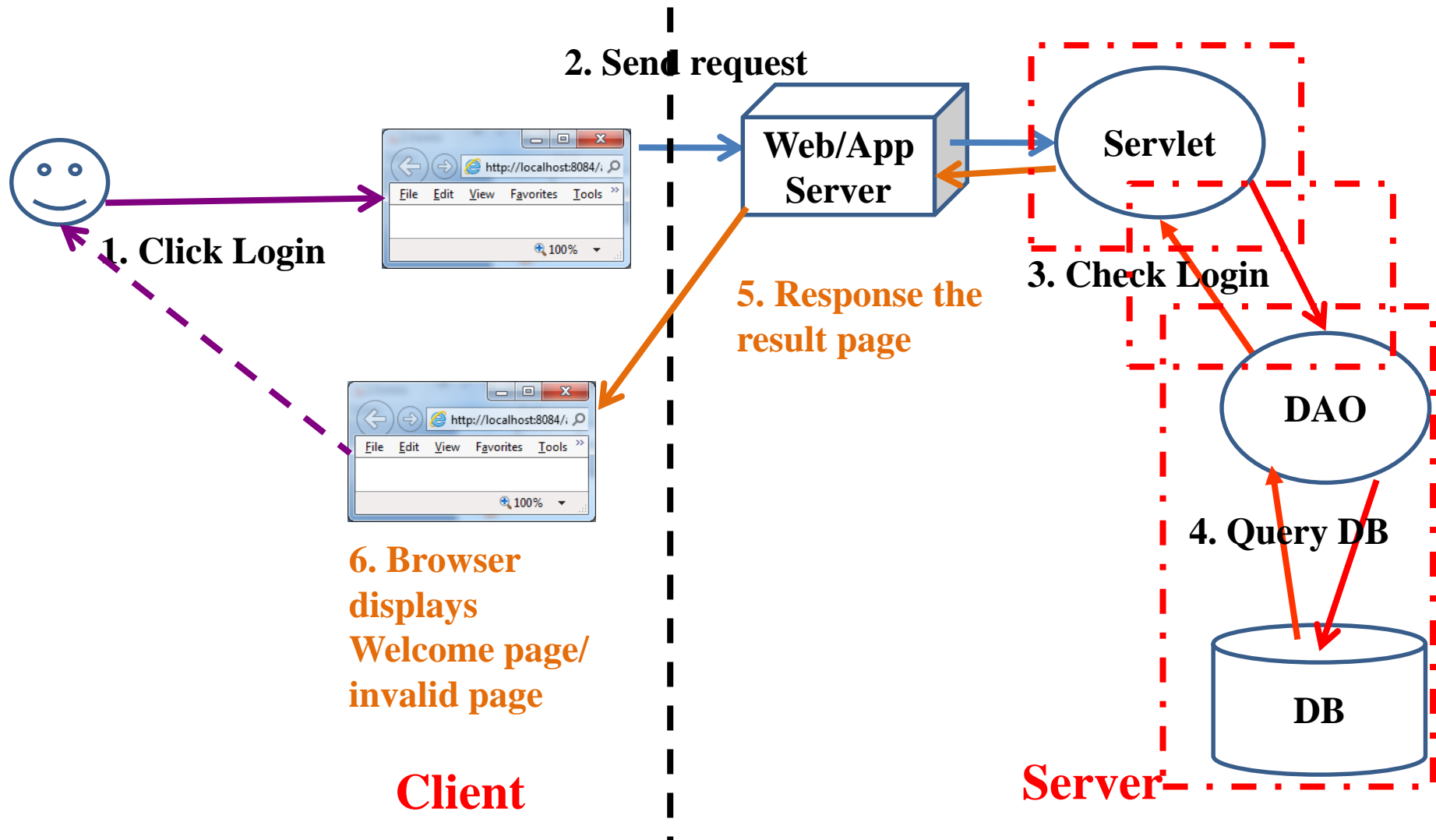
The screenshot shows the 'Servers' view on the left with a context menu open for 'AJDay1_7', highlighting 'Undeploy'. The 'Projects' view on the right shows a context menu for 'AJDay1_7' with 'Clean and Build' highlighted. Below these, a console window shows the output of the undeploy action.

```

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

Build The Simple Web

Interactive Server Model

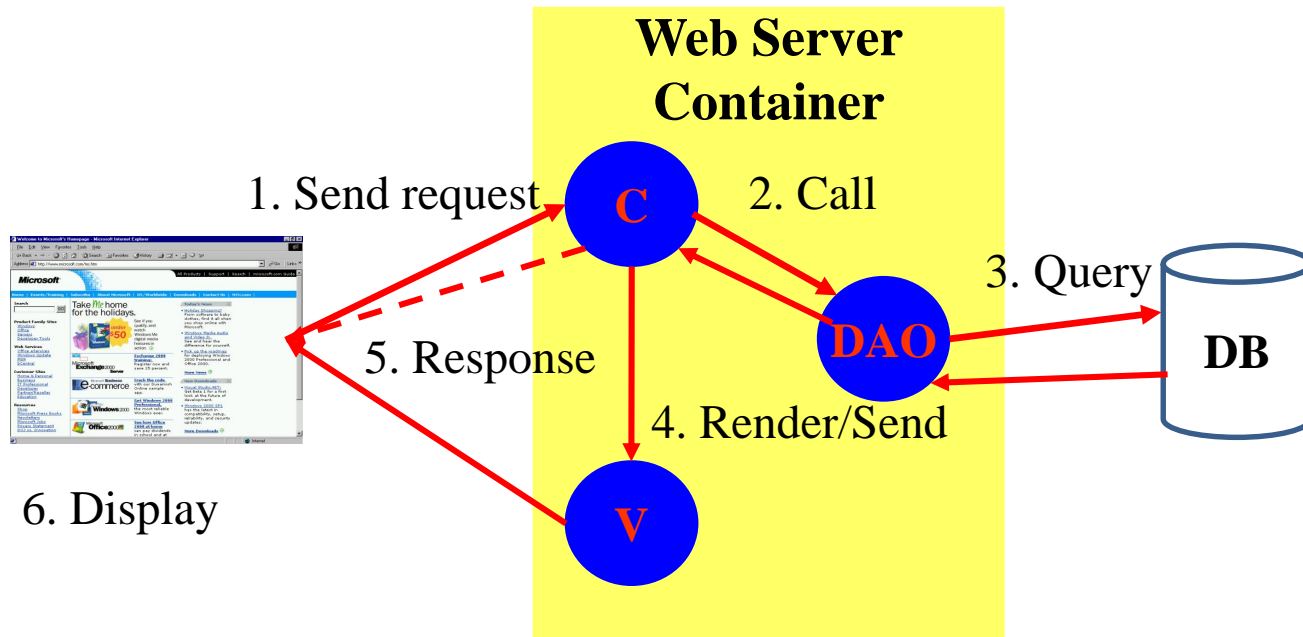


Summary

- **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



Q&A

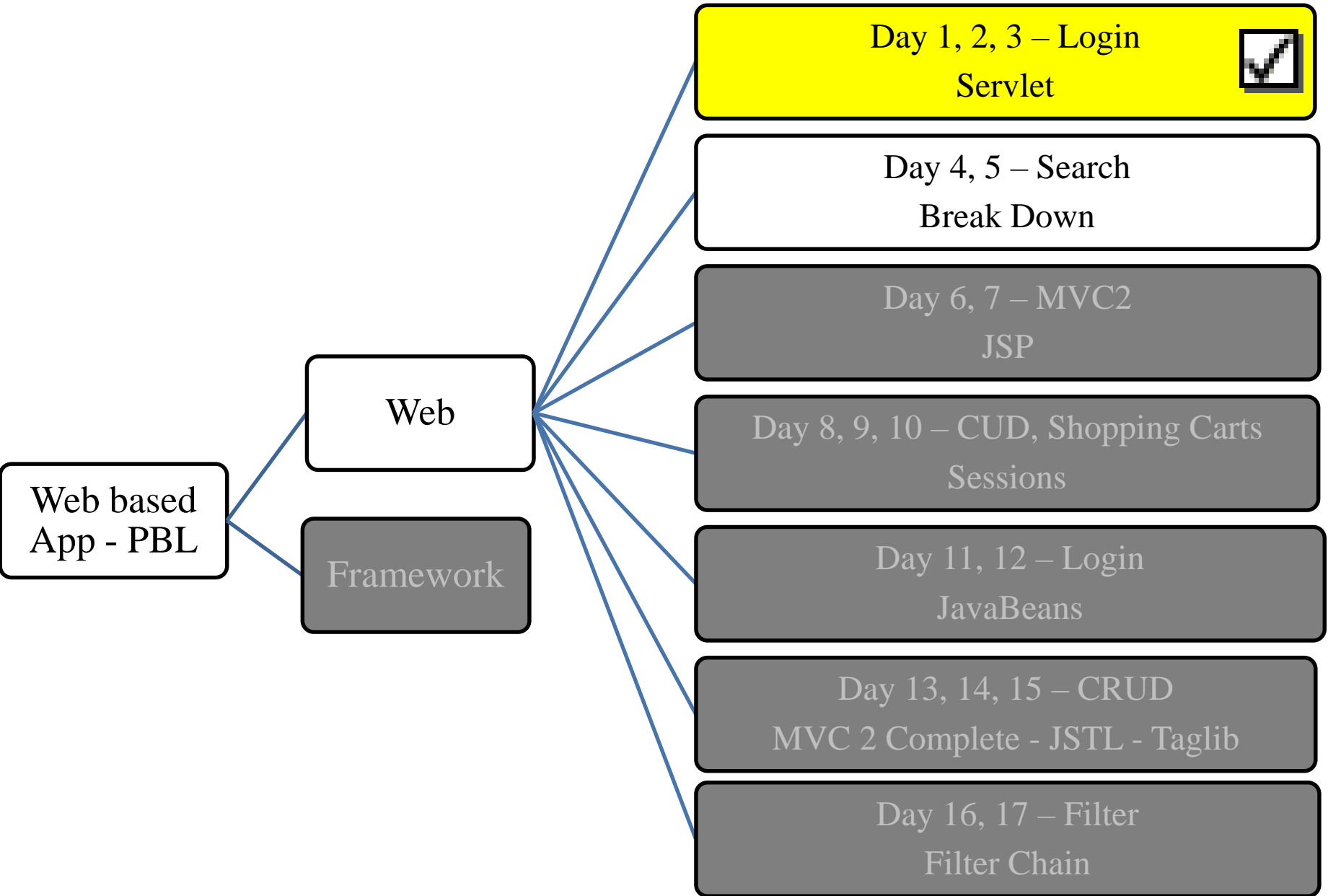
Exercises

- 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

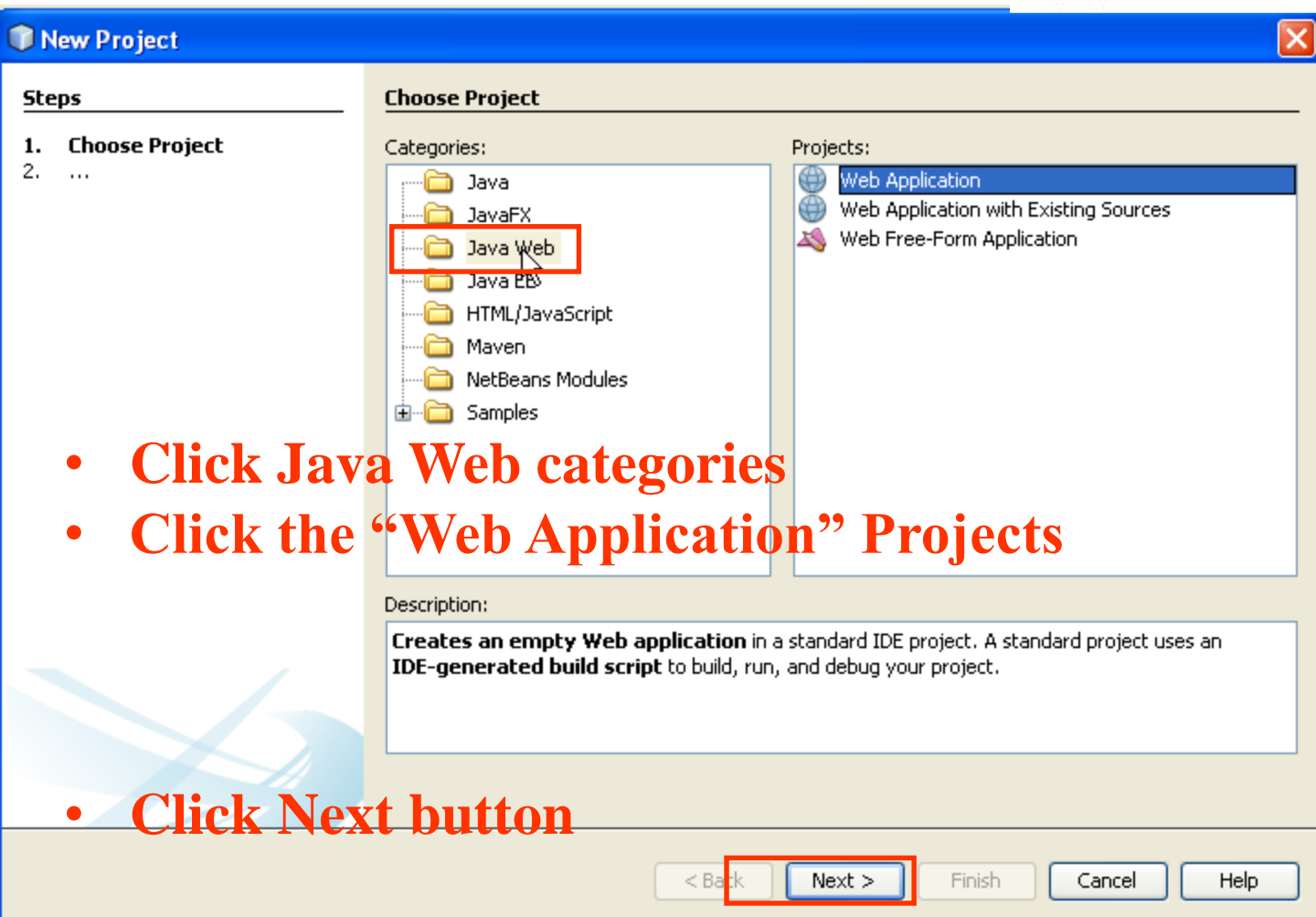
- **How to deploy the Web Application to Web Server?**
 - Web applications Structure
 - Request Parameters vs. Context Parameters vs. Config/Servlet Parameters
 - Application Segments vs. Scope
- **How to transfer from resources to others with/without data/objects?**
 - Attributes vs. Parameters vs.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

Next Lecture



Appendix – Build The Simple Web

How to Create Web Application 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:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

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

Buttons: < Back, **Next >**, Finish, Cancel, Help

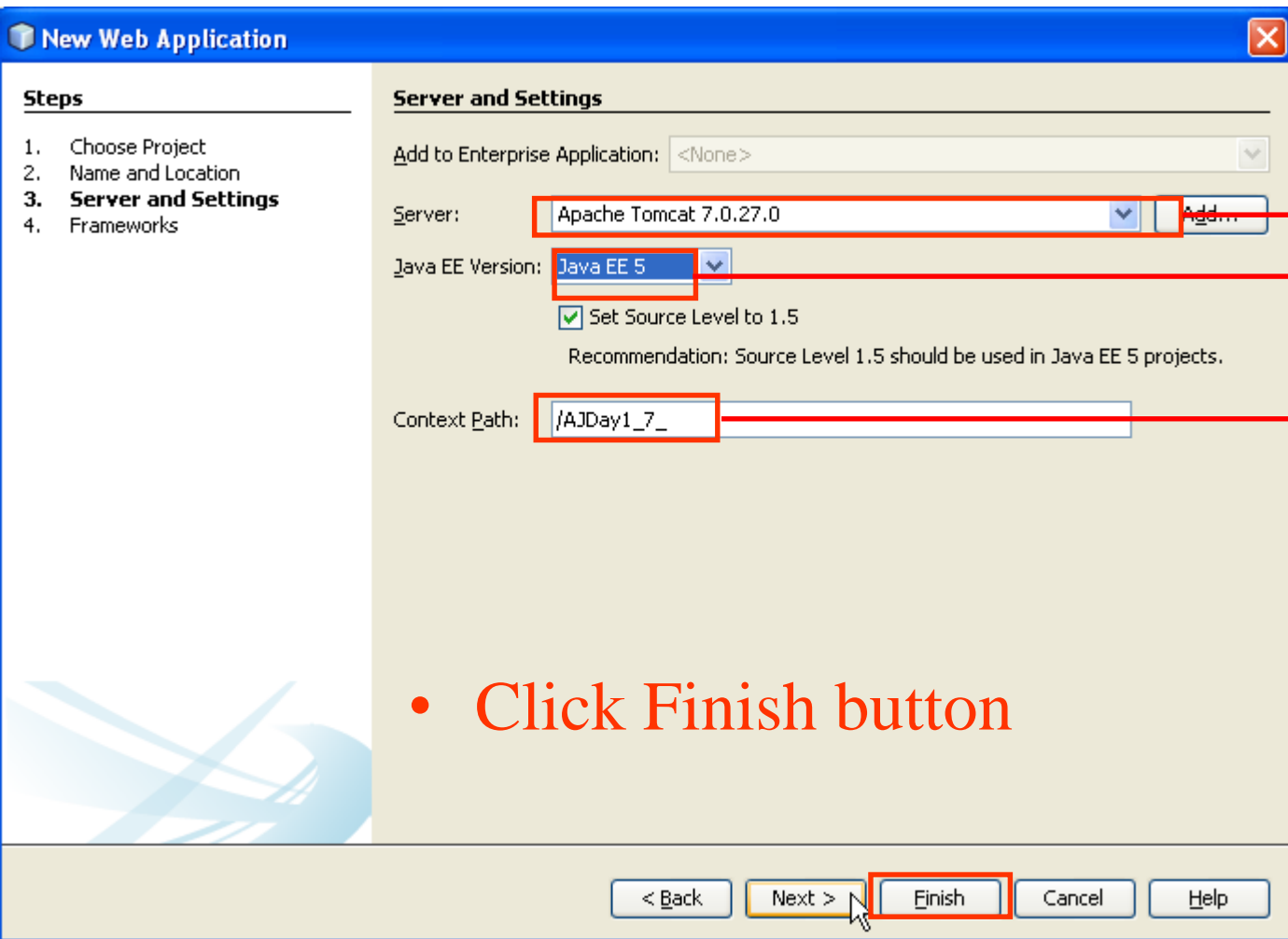
Fill your project name

Browse 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_

< Back Next > **Finish** Cancel Help

Choose deployed server

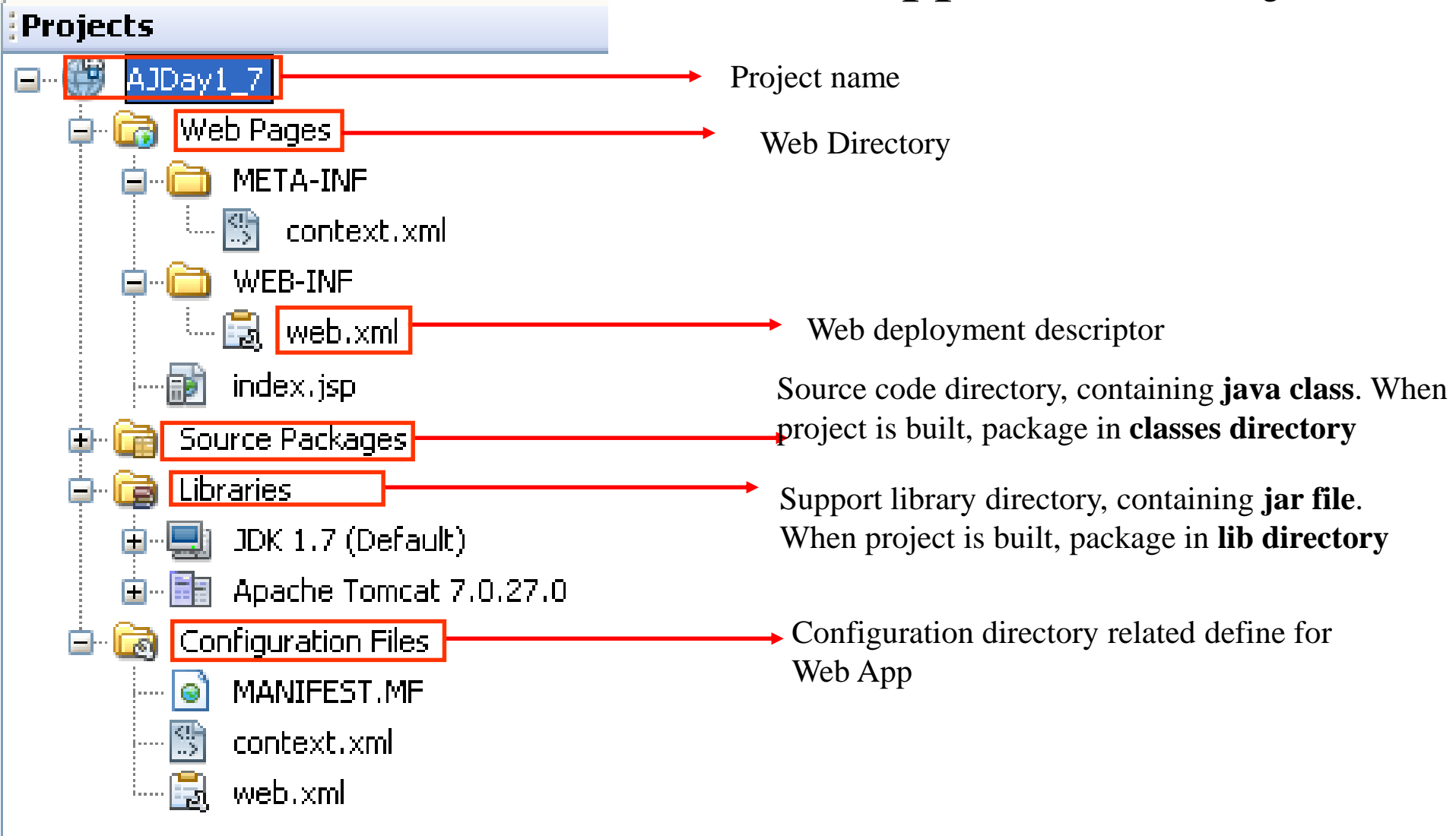
Choose Java EE 5/
J2EE 1.4

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

- Click Finish button

Appendix – Build The Simple Web

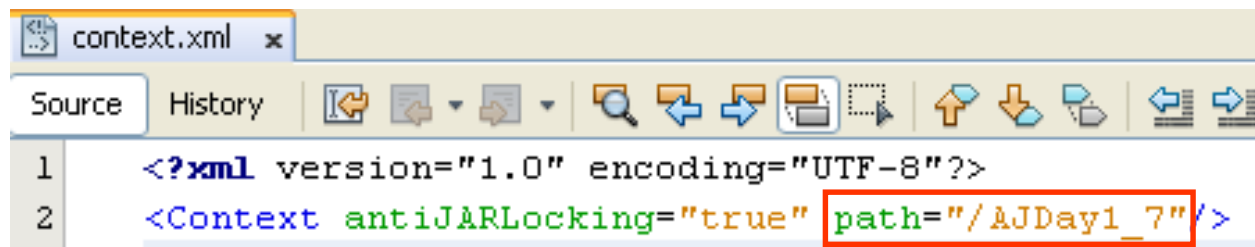
How to Create Web Application Project



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)

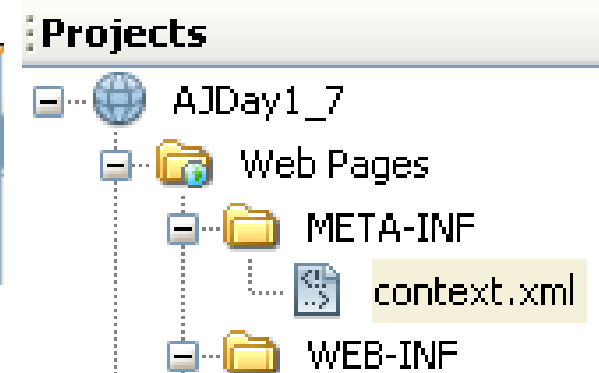


The screenshot shows an XML editor with a toolbar and a source view. The file is named 'context.xml'. The content is as follows:

```

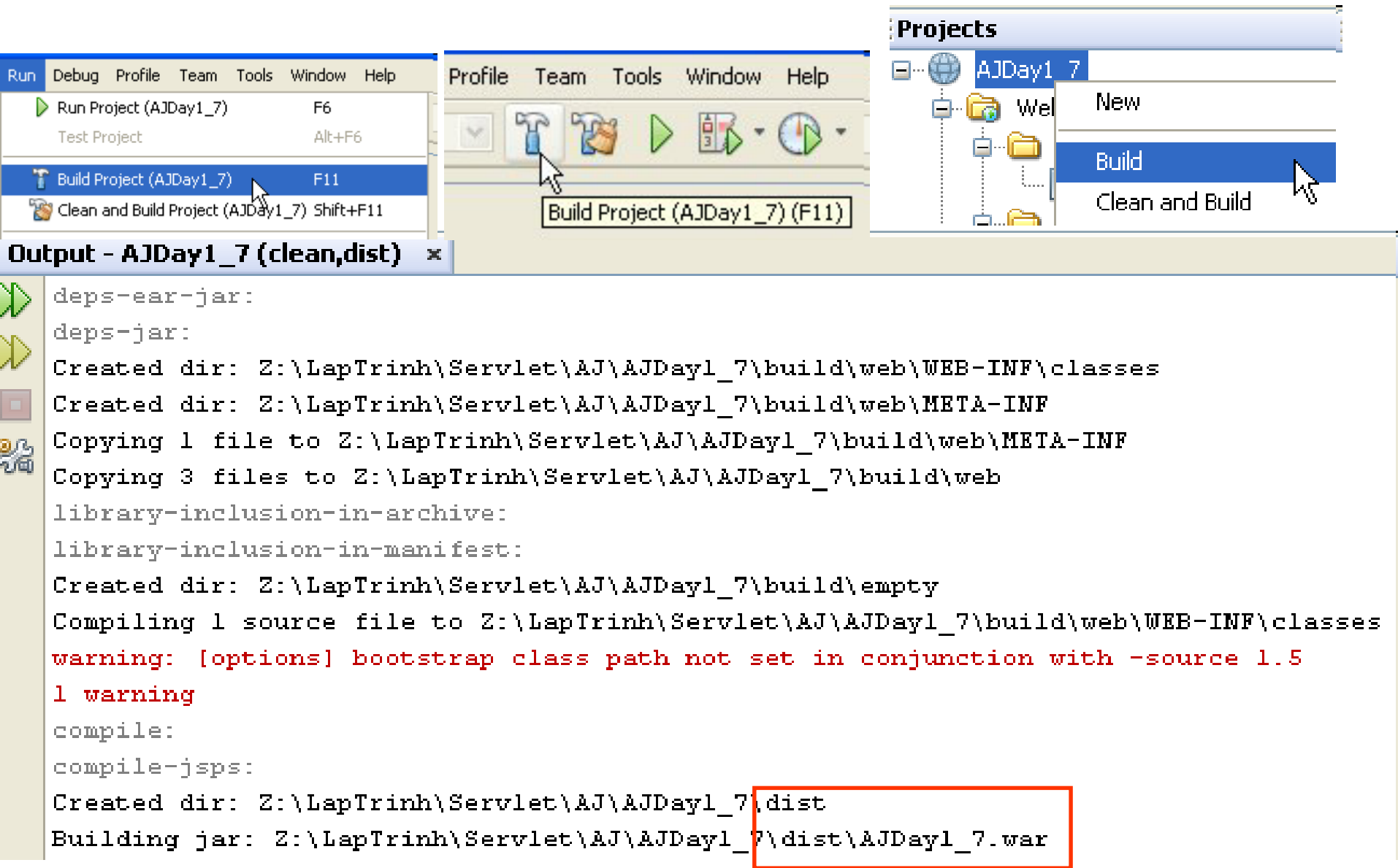
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Context antiJARLocking="true" path="/AJDay1_7"/>
  
```

The 'path' attribute in the second line is highlighted with a red rectangle.



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

Profile Team Tools Window Help

Build Project (AJDay1_7) (F11)

Projects

- AJDay1_7
 - New
 - Build**
 - Clean and Build

Output - AJDay1_7 (clean,dist)

```

>> 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-jsp:
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\AJDay1_7.war*			
↑Name	Ext	Size	Date
↑ [..]		<DIR>	18/0
AJDay1_7	war	4.183	18/0

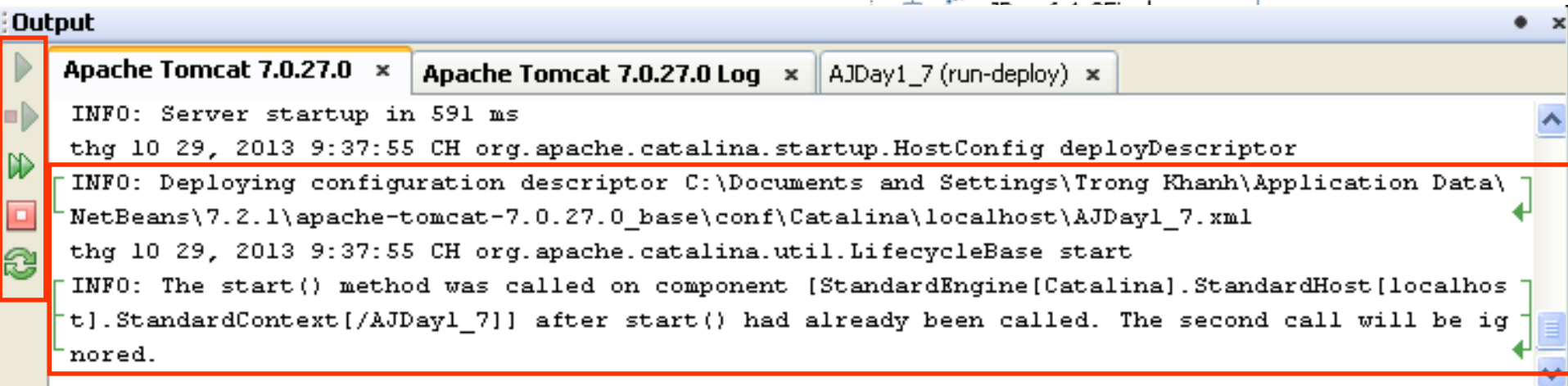
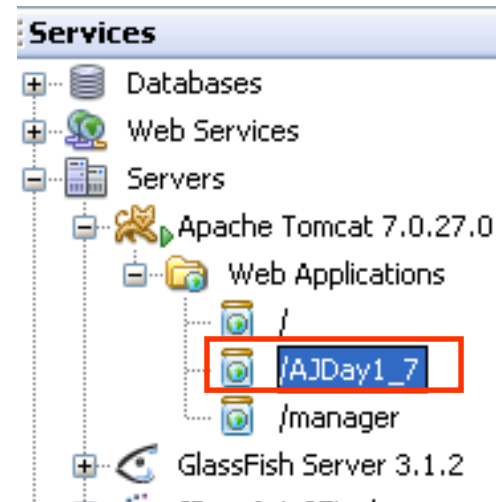
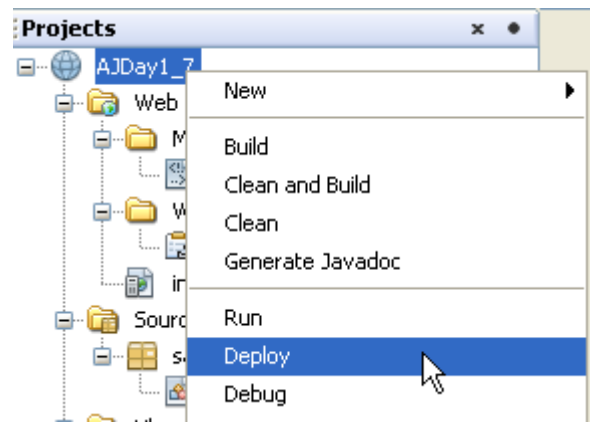
z:\LapTrinh\Servlet\AJ\AJDay1_7\build\web*		
↑Name	Ext	Size
↑ [..]		<DIR>
[META-INF]		<DIR>
[WEB-INF]		<DIR>
index	jsp	372

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/***

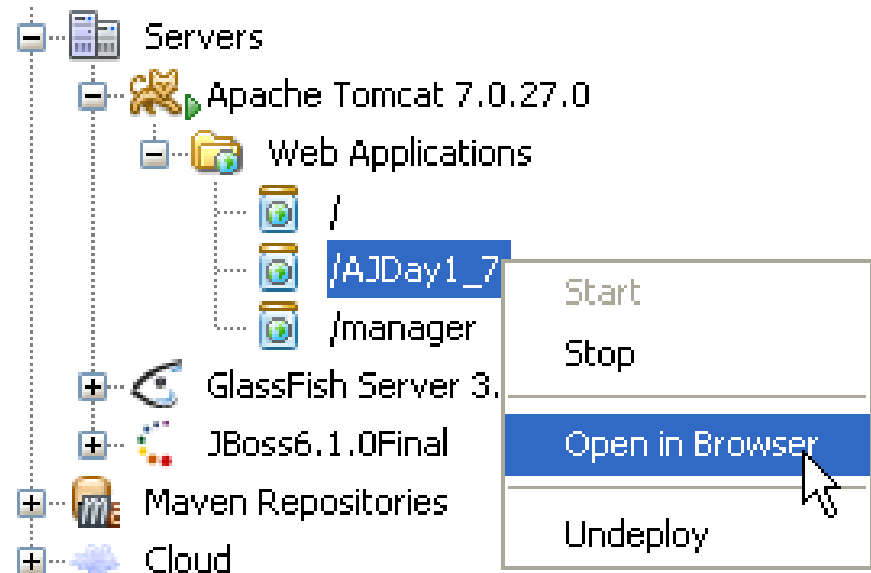
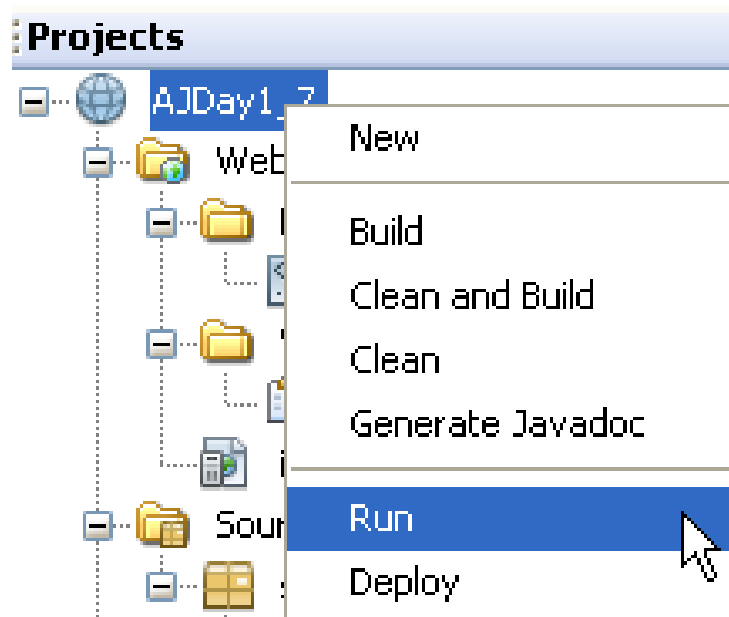
Appendix

Deploy Application



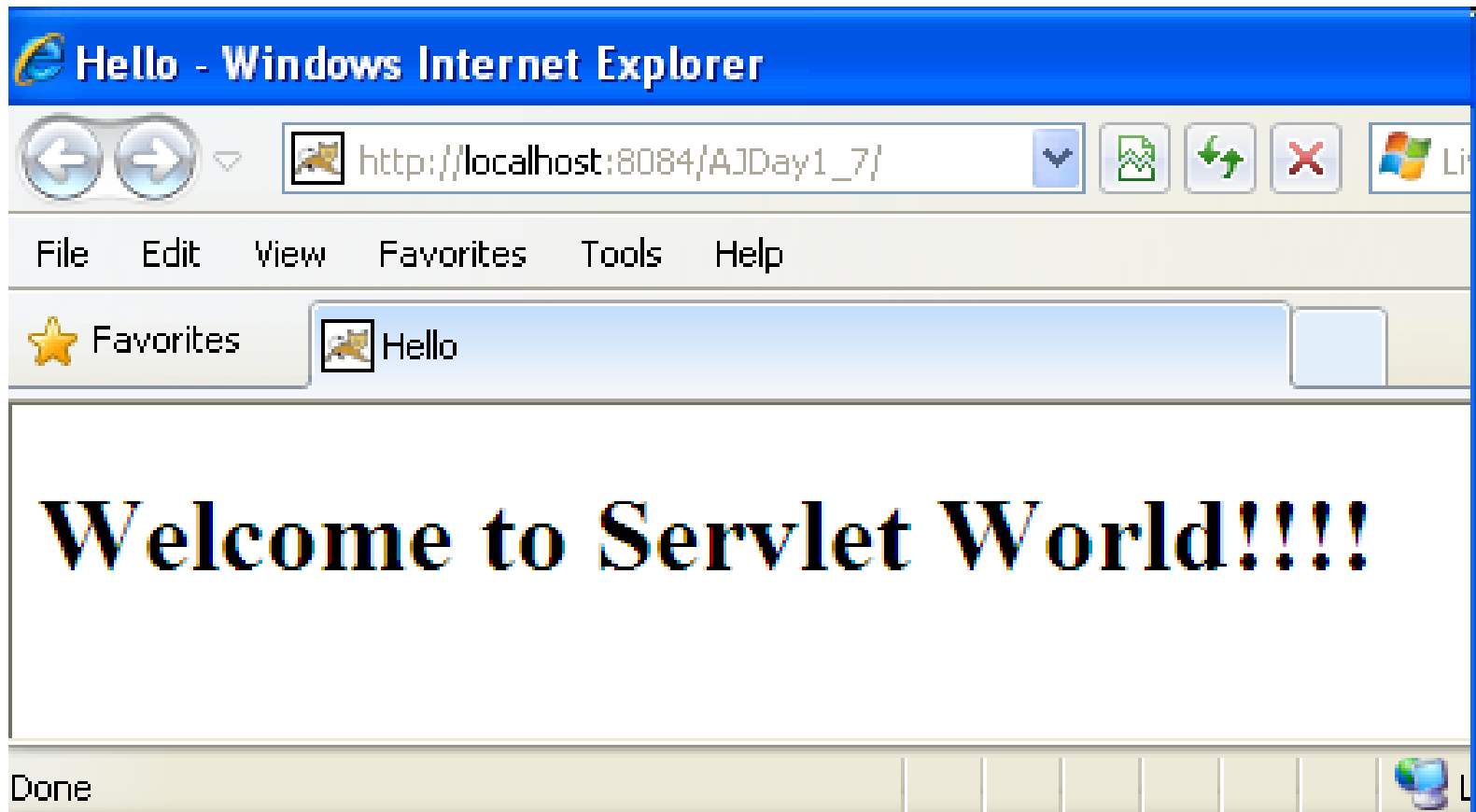
Appendix

Run Application



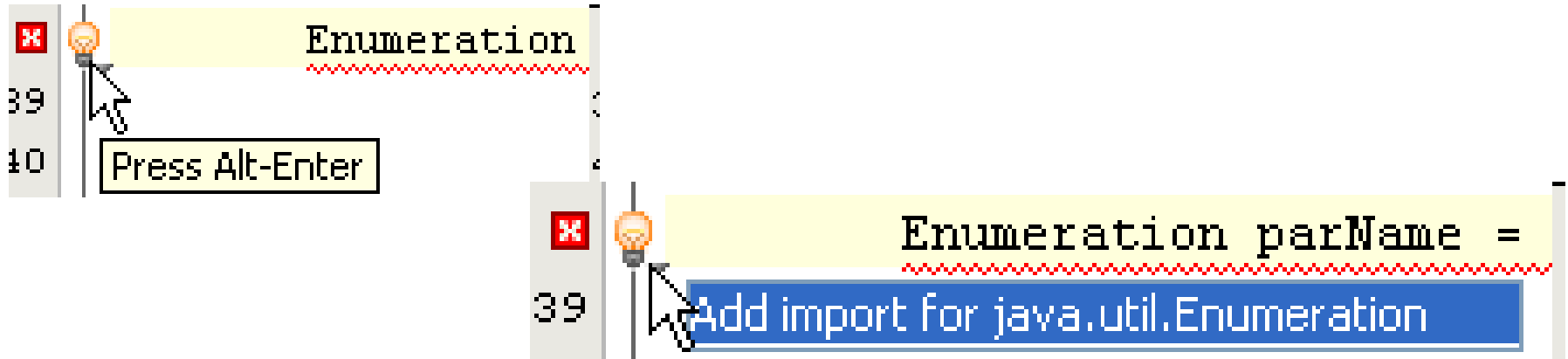
Appendix

Run Application



Appendix

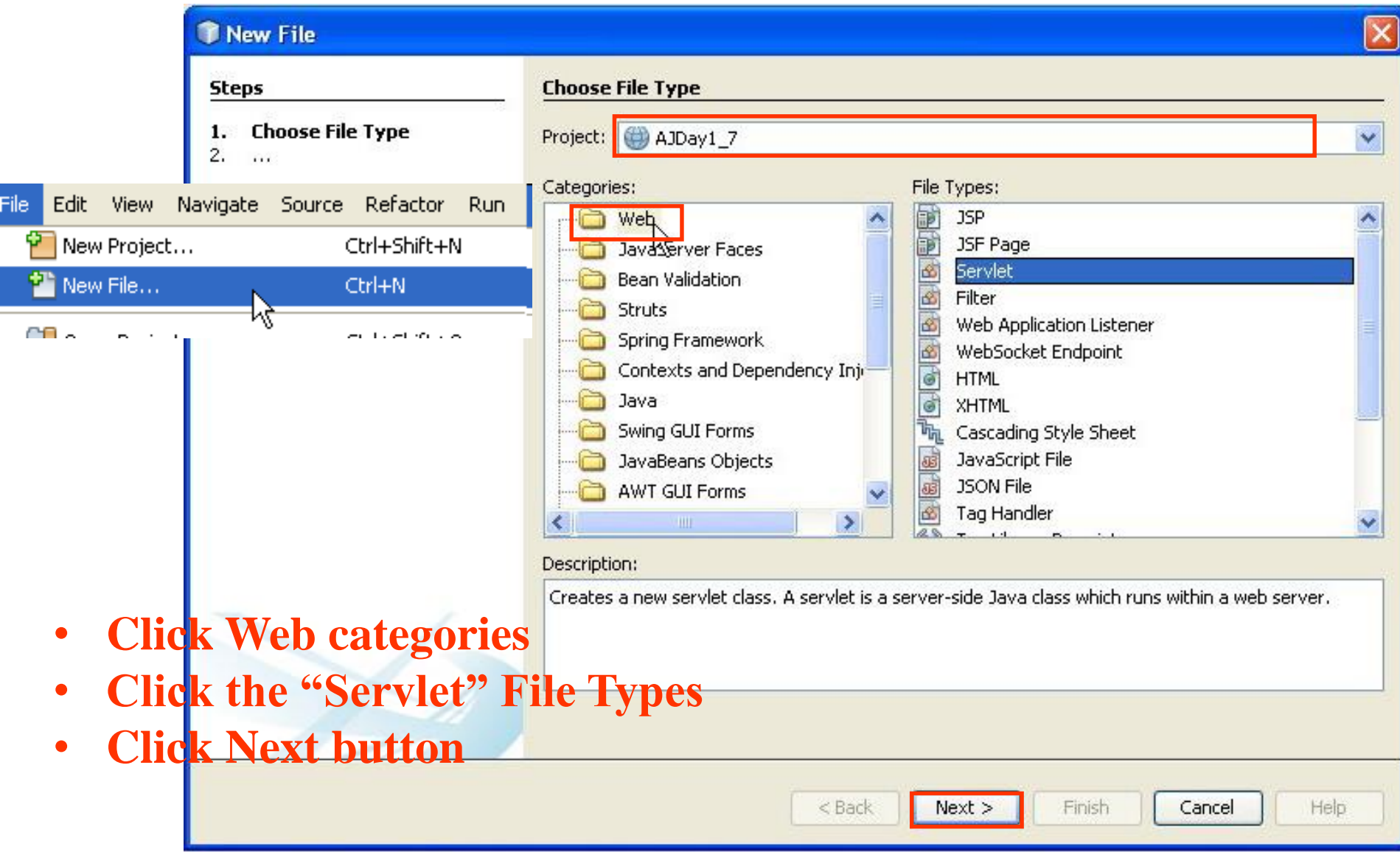
Additional



- **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



- Click Web categories
- Click the “Servlet” File Types
- Click Next button

Appendix

Create a Servlet

New Servlet

Steps

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

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

Buttons: < Back, **Next >**, Finish, Cancel, Help

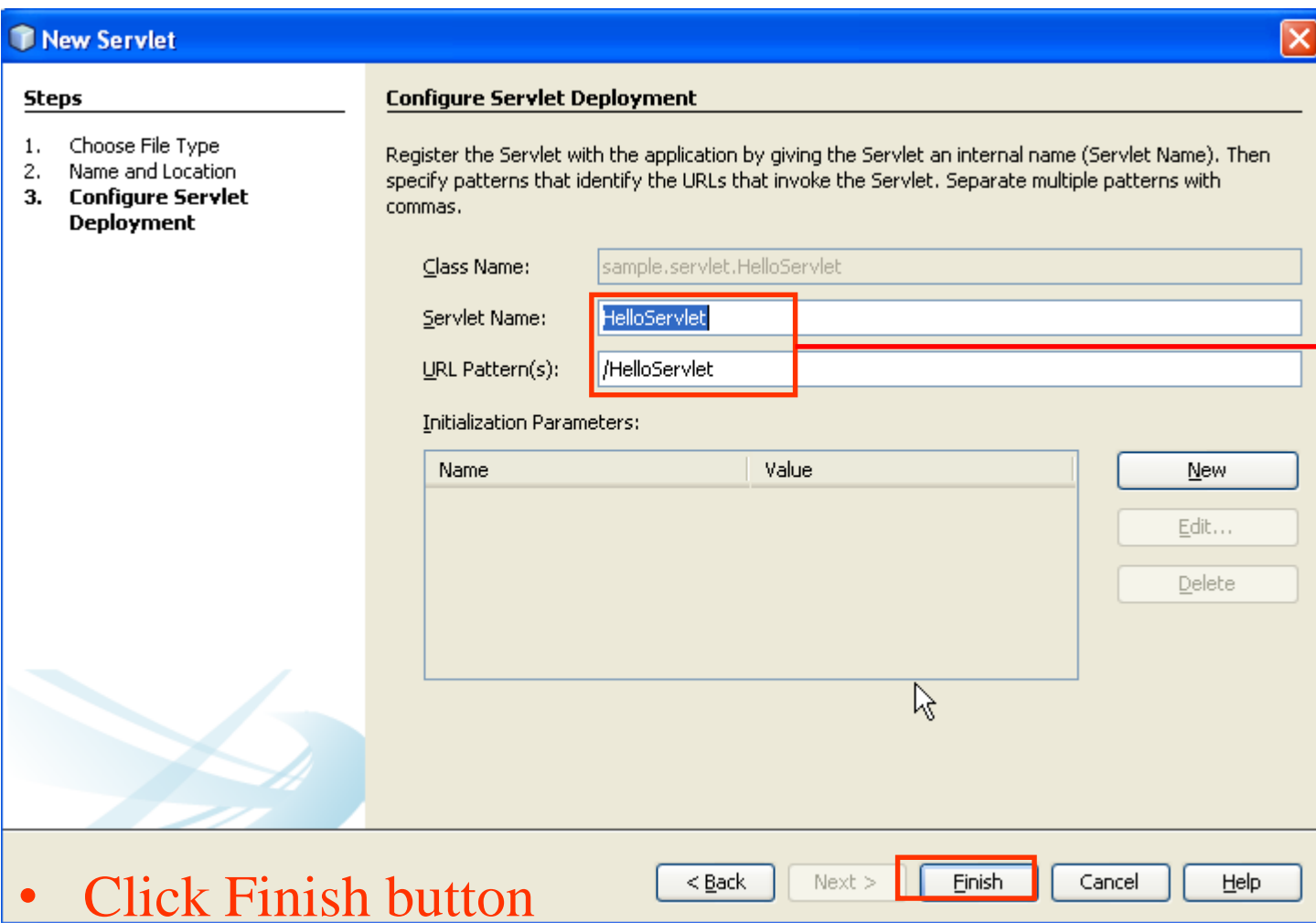
• Click Next button

Fill your servlet name

Fill or choose package name

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:

Servlet Name:

URL Pattern(s):

Initialization Parameters:

Name	Value
------	-------

New Edit... Delete

< Back Next > **Finish** Cancel Help

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

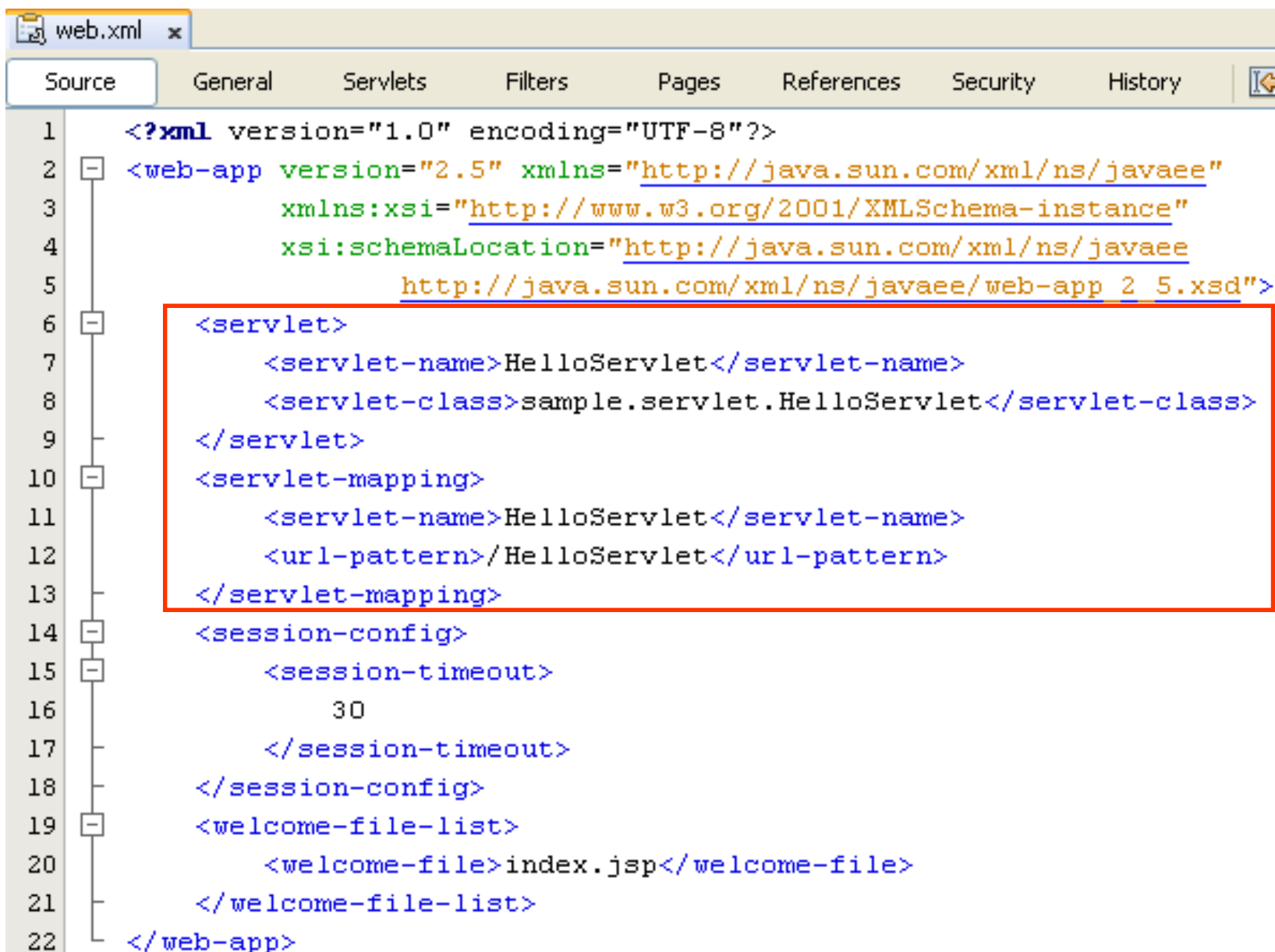
```

HelloServlet.java x
Source History
15
16 * @author Trong Khanh
17 */
18 public class HelloServlet extends HttpServlet {
19
20     /** ... */
30     protected void processRequest(HttpServletRequest request,
31         HttpServletResponse response)
32         throws ServletException, IOException {
33         response.setContentType("text/html;charset=UTF-8");
34         PrintWriter out = response.getWriter();
35         try {
36             /* TODO output your page here. You may use following s
37             out.println("<!DOCTYPE html>");
38             out.println("<html>");
39             out.println("<head>");
40             out.println("<title>Hello</title>");
41             out.println("</head>");
42             out.println("<body>");
43             out.println("<h1>Welcome to Servlet World!!!!</h1>");
44             out.println("</body>");
45             out.println("</html>");
46         } finally {
47             out.close();
48         }
49     }
50
51     HttpServlet methods. Click on the + sign on the left to edit t
91 }

```

Appendix

Create a Servlet



```

web.xml x
Source General Servlets Filters Pages References Security History
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5         http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
6     <servlet>
7         <servlet-name>HelloServlet</servlet-name>
8         <servlet-class>sample.servlet.HelloServlet</servlet-class>
9     </servlet>
10    <servlet-mapping>
11        <servlet-name>HelloServlet</servlet-name>
12        <url-pattern>/HelloServlet</url-pattern>
13    </servlet-mapping>
14    <session-config>
15        <session-timeout>
16            30
17        </session-timeout>
18    </session-config>
19    <welcome-file-list>
20        <welcome-file>index.jsp</welcome-file>
21    </welcome-file-list>
22 </web-app>
  
```

The Servlet Model

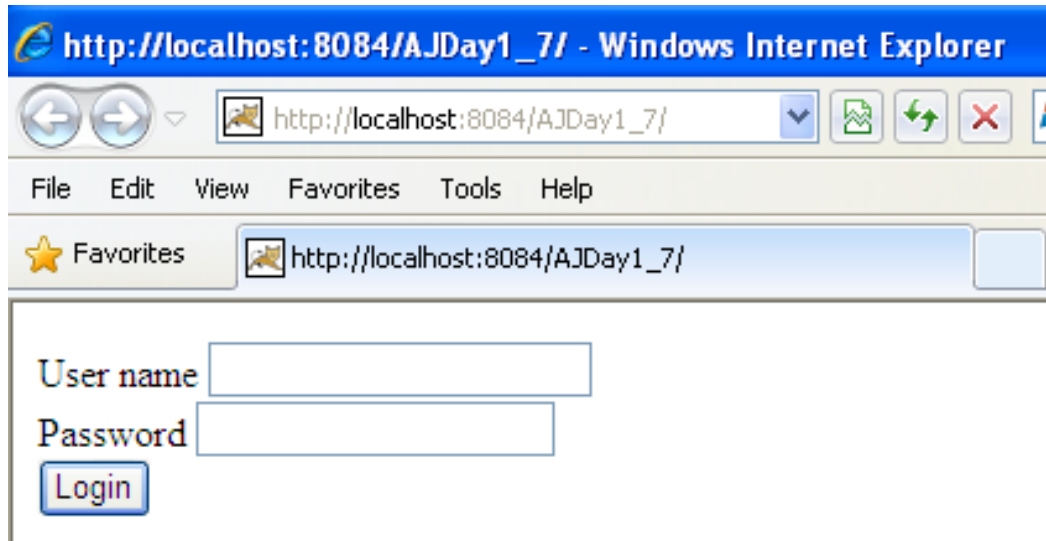
ServletRequest interface

- 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	<ul style="list-style-type: none"> - public String getParameter(String name) - Returns the value of a specified parameter by the name (or null or “”) - String strUser = request.getParameter(“txtUser”);
getParameterNames	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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

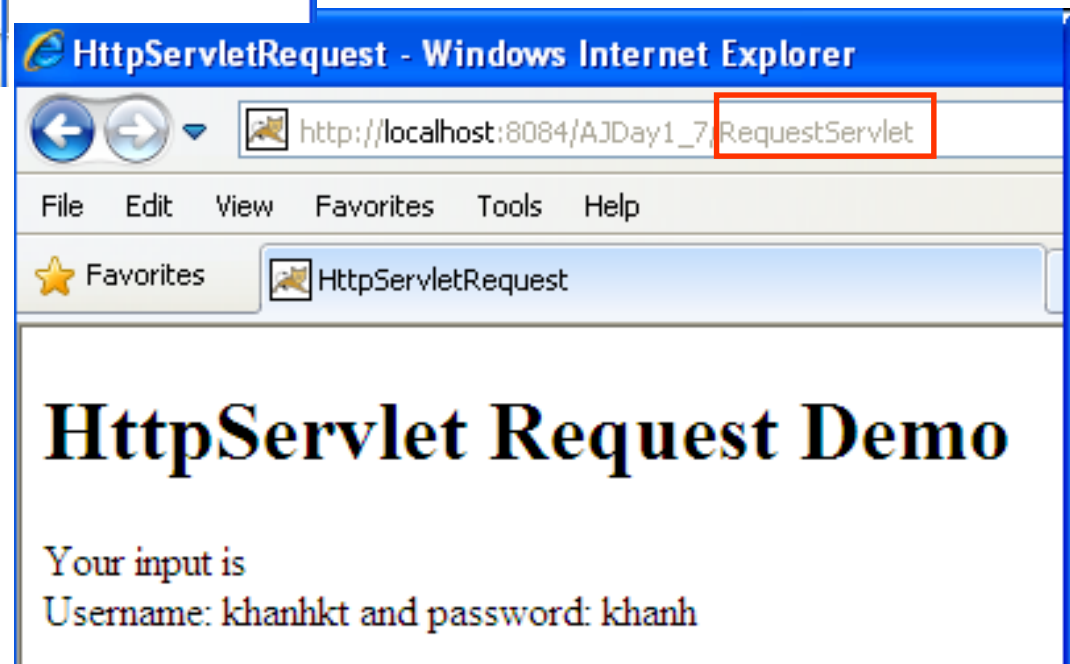
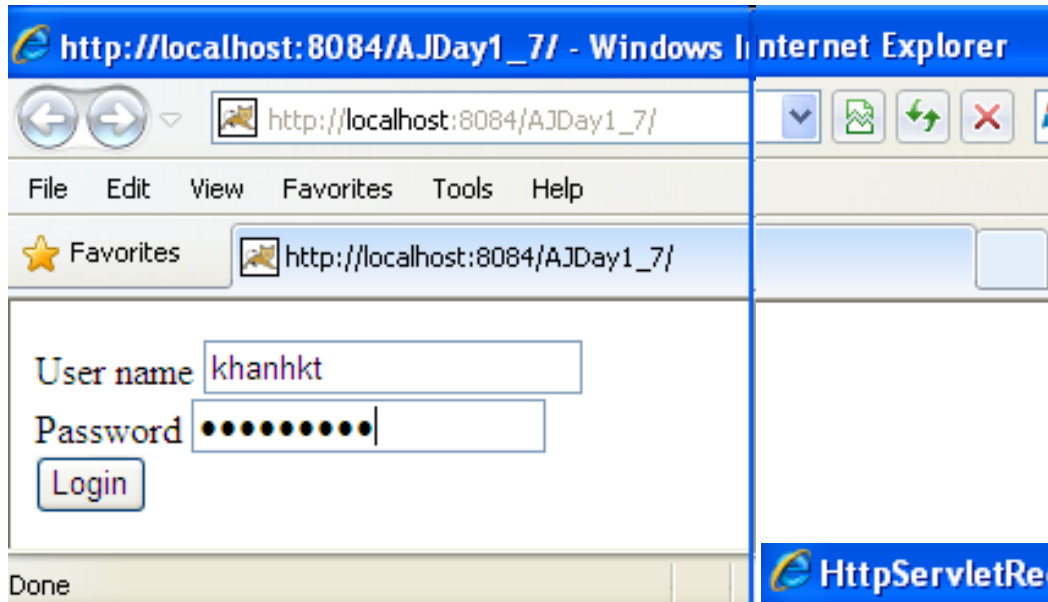
HttpServletRequest interface – Examples



The screenshot shows a Windows Internet Explorer browser window. The address bar displays the URL `http://localhost:8084/AJDay1_7/`. The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. Below the menu bar, there is a Favorites bar showing the same URL. The main content area of the browser displays a simple login form with two text input fields. The first field is labeled "User name" and the second is labeled "Password". Below these fields is a button labeled "Login".

Appendix – Servlet Model

HttpServletRequest interface – Examples



Appendix – The Servlet Model

HttpServletRequest interface – Examples

```

1  ...
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
6  <html>
7      <head>
8          <title></title>
9          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10     </head>
11     <body>
12         <form action="RequestServlet" method="post">
13             User name <input type="text" name="txtUser"/><br/>
14             Password <input type="password" name="txtPass"/><br/>
15             <input type="submit" value="Login"/><br/>
16         </form>
17     </body>
18 </html>
  
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples

```

RequestServlet.java x
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>HttpServletRequest 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

Password

HttpServletRequest - Windows Internet Explorer

http://localhost:8084/AJDay1_7/RequestServlet

File Edit View Favorites Tools Help

★ Favorites HttpServletRequest

HttpServletRequest 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

```

RequestServlet.java x
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>HttpServletRequest 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/

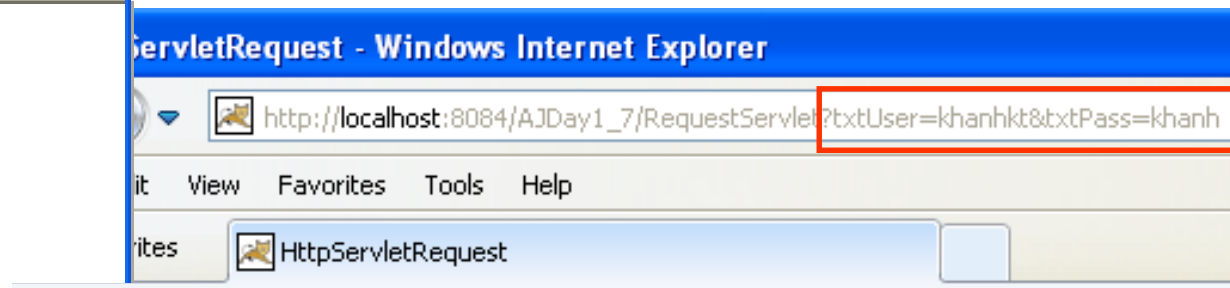
File Edit View Favorites Tools Help

Favorites http://localhost:8084/AJDay1_7/

User name

Password

Login



HttpServletRequest 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

```

1  ...
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
6  <html>
7      <head>
8          <title></title>
9          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10     </head>
11     <body>
12         <form action="RequestServlet">
13             User name <input type="text" name="txtUser"/><br/>
14             Password <input type="password" name="txtPass"/><br/>
15             <input type="submit" value="Login"/><br/>
16         </form>
17     </body>
18 </html>
  
```

Appendix – The Servlet Model

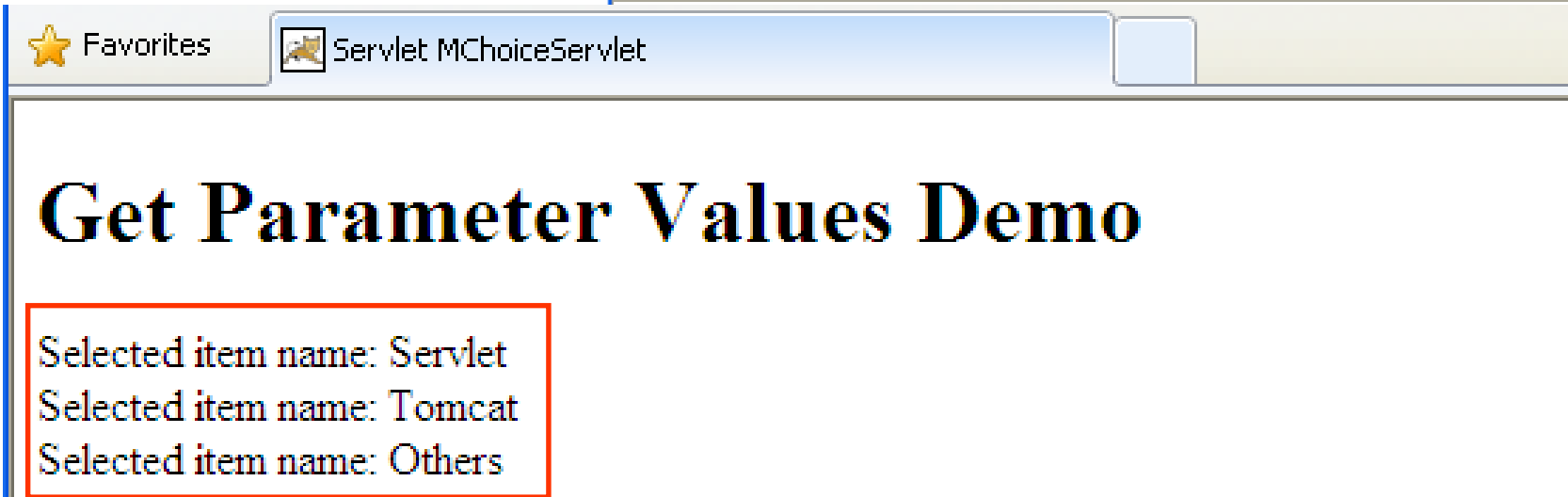
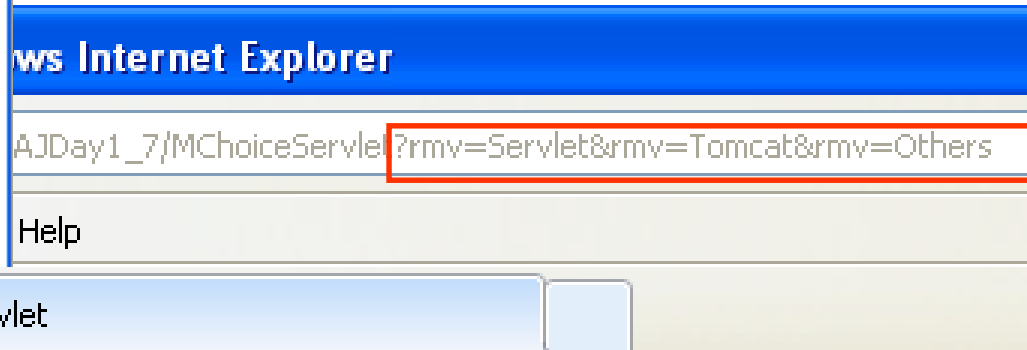
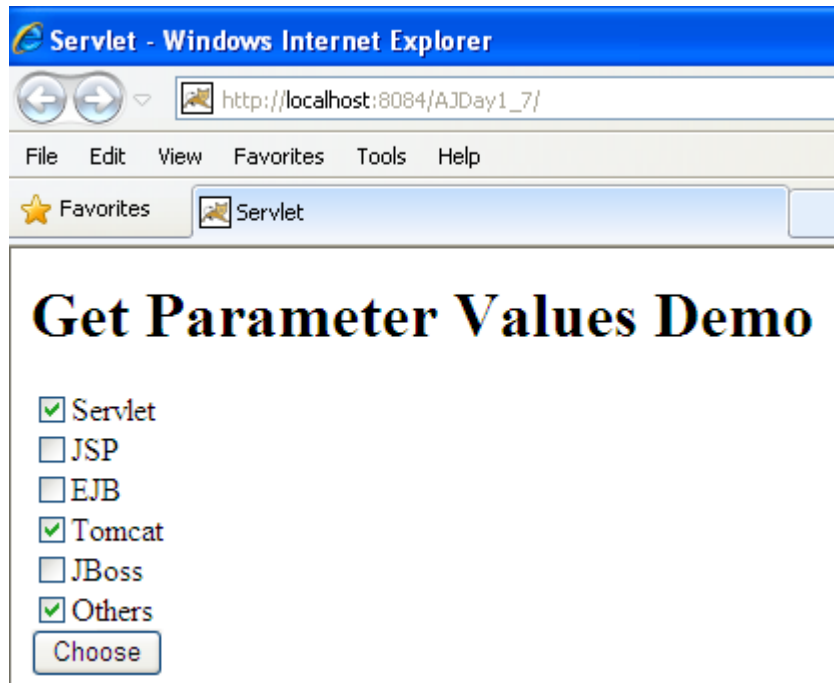
HttpServletRequest interface – Examples

```

RequestServlet.java x
36
37 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
38     throws ServletException, IOException {
39     response.setContentType("text/html;charset=UTF-8");
40     PrintWriter out = response.getWriter();
41     try {
42         out.println("<html>");
43         out.println("<head>");
44         out.println("<title>Servlet RequestServlet</title>");
45         out.println("</head>");
46         out.println("<body>");
47         out.println("<h1>HttpServletRequest Demo</h1>");
48         Enumeration parNames = request.getParameterNames();
49         int count = 0;
50         while (parNames.hasMoreElements()) {
51             ++count;
52             String parName = (String) parNames.nextElement();
53             out.print("parName" + count + " is " + parName);
54             String parVal = request.getParameter(parName);
55             out.println(" and value is " + parVal + "<br/>");
56         }
57         String strServer = request.getServerName();
58         out.println("Server Name: " + strServer + "<br/>");
59         String strHost = request.getHeader("host");
60         out.println("Header - host: " + strHost + "<br/>");
61         String strMethod = request.getMethod();
62         out.println("Request Method " + strMethod + "<br/>");
63         String qs = request.getQueryString();
64         out.println("Query String " + qs + "<br/><br/>");
65         out.println("</body>");
66         out.println("</html>");
  
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples



Appendix – The Servlet Model

HttpServletRequest interface – Examples

```

parameterValues.html x
Preview
1  ...
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
6  <html>
7  <head>
8      <title>Servlet</title>
9      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10 </head>
11 <body>
12     <h1>Get Parameter Values Demo</h1>
13     <form action="MChoiceServlet">
14         <input type="checkbox" name="rmv" value="Servlet" />Servlet<br/>
15         <input type="checkbox" name="rmv" value="JSP" />JSP<br/>
16         <input type="checkbox" name="rmv" value="EJB" />EJB<br/>
17         <input type="checkbox" name="rmv" value="Tomcat" />Tomcat<br/>
18         <input type="checkbox" name="rmv" value="JBoss" />JBoss<br/>
19         <input type="checkbox" name="rmv" value="Others" />Others<br/>
20         <input type="submit" value="Choose" />
21     </form>
22 </body>
23 </html>
  
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples

```

MChoiceServlet.java x
26
27     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
28         throws ServletException, IOException {
29         response.setContentType("text/html;charset=UTF-8");
30         PrintWriter out = response.getWriter();
31         try {
32             out.println("<html>");
33             out.println("<head>");
34             out.println("<title>Servlet MChoiceServlet</title>");
35             out.println("</head>");
36             out.println("<body>");
37             out.println("<h1>Get Parameter Values Demo</h1>");
38             String[] strSelect = request.getParameterValues("rmv");
39             if (strSelect != null) {
40                 for (int i = 0; i < strSelect.length; i++) {
41                     out.println("Selected item name: " + strSelect[i] + "<br/>");
42                 }
43             }
44             out.println("</body>");
45             out.println("</html>");
46         } finally {
47             out.close();
48         }
49     }
  
```

Appendix – The Servlet Model

HttpServletRequest interface – Examples

```
<body>
<form action="Controller">
```

```
Num1 <input type="text" name="txtNum"/> <br/>
```

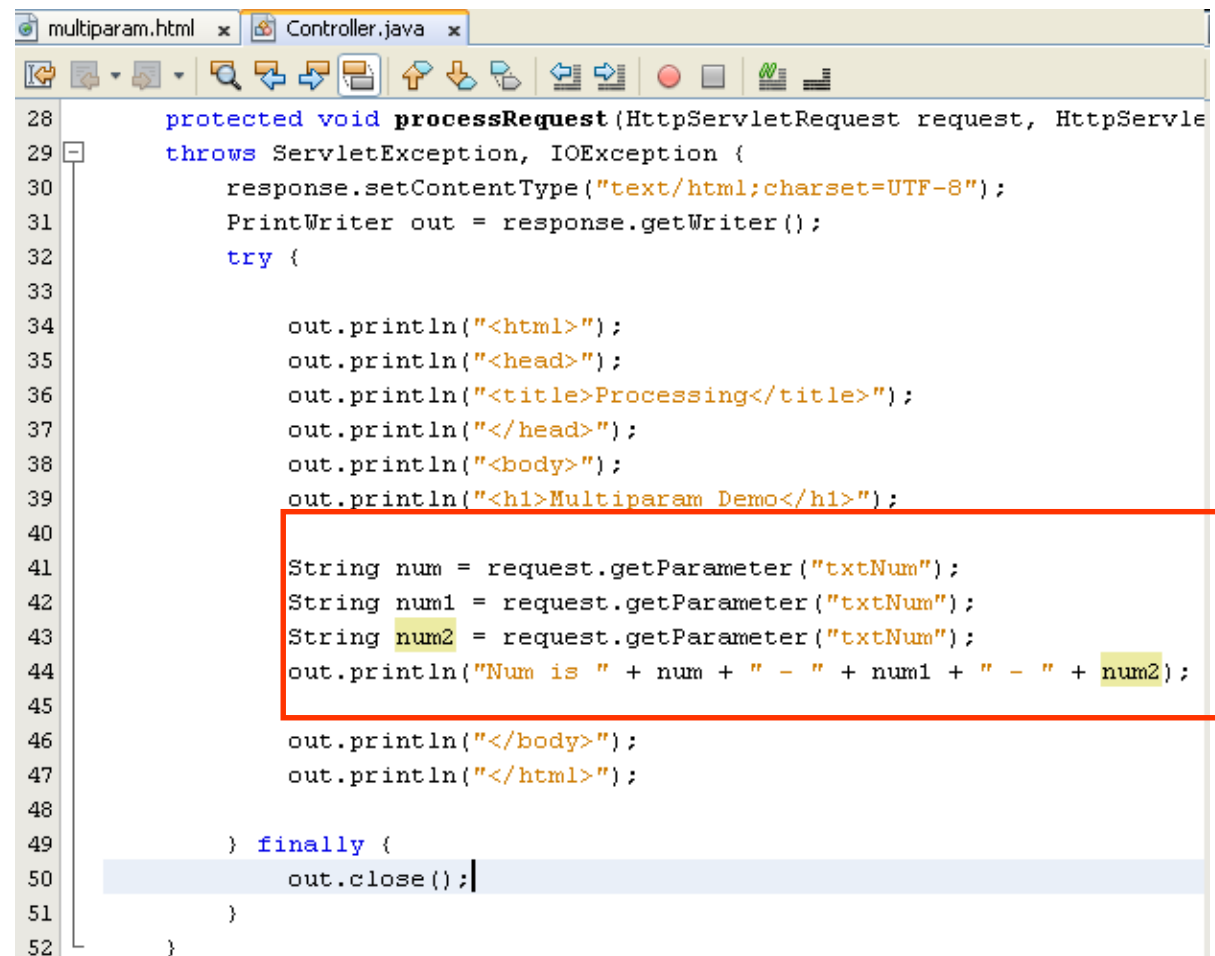
```
Num2 <input type="text" name="txtNum"/> <br/>
```

```
Num3 <input type="text" name="txtNum"/> <br/>
```

```
<input type="submit" value="Perform" />
```

```
</form>
```

```
</body>
```



```

28     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
29         throws ServletException, IOException {
30         response.setContentType("text/html;charset=UTF-8");
31         PrintWriter out = response.getWriter();
32         try {
33
34             out.println("<html>");
35             out.println("<head>");
36             out.println("<title>Processing</title>");
37             out.println("</head>");
38             out.println("<body>");
39             out.println("<h1>Multiparam Demo</h1>");
40
41             String num = request.getParameter("txtNum");
42             String num1 = request.getParameter("txtNum");
43             String num2 = request.getParameter("txtNum");
44             out.println("Num is " + num + " - " + num1 + " - " + num2);
45
46             out.println("</body>");
47             out.println("</html>");
48
49         } finally {
50             out.close();
51         }
52     }

```


Appendix – The Servlet Model

HttpServletRequest interface – Examples

MultiParam - Windows Internet Explorer

http://localhost:8084/AJDay1_7/

File Edit View Favorites Tools Help

Favorites MultiParam

Multiple Parameter Demo

Num1

Num2

Num3

Windows Internet Explorer

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

Favorites Tools Help

Favorites Processing

Multipleparam Demo

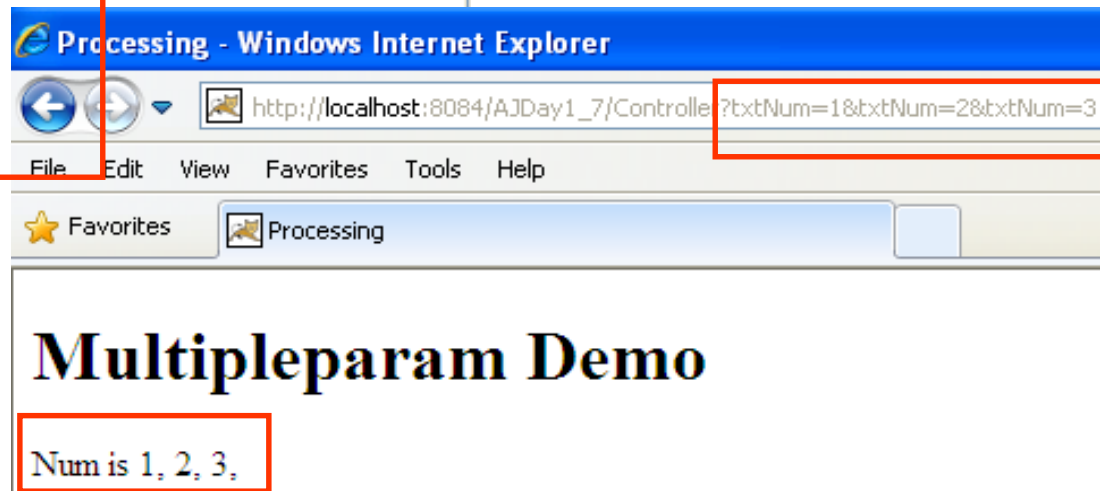
Num is

HttpServletRequest interface – Examples

```

28  protected void processRequest(HttpServletRequest request, Http
29  throws ServletException, IOException {
30      response.setContentType("text/html;charset=UTF-8");
31      PrintWriter out = response.getWriter();
32      try {
33
34          out.println("<html>");
35          out.println("<head>");
36          out.println("<title>Processing</title>");
37          out.println("</head>");
38          out.println("<body>");
39          out.println("<h1>Multiparam Demo</h1>");
40
41          String[] num = request.getParameterValues("txtNum");
42          out.println("Num is: ");
43          for(int i=0; i<num.length; i++) {
44              out.println(num[i] + ", ");
45          }
46
47          out.println("</body>");
48          out.println("</html>");
49
50      } finally {
51          out.close();
52      }
53  }

```



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	<ul style="list-style-type: none">- 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	<ul style="list-style-type: none">- 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	<ul style="list-style-type: none"> - public ServletOutputStream getOutputStream() throws IOException - Uses ServletOutputStream object to write response as binary data to the client. - ServletOutputStream out = response.getOutputStream(); - 02 supporting methods <ul style="list-style-type: none"> + public void print(boolean b) throws IOException <ul style="list-style-type: none"> . 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 <ul style="list-style-type: none"> . same as the print methods but it writes a character value to the client, followed by a carriage return line feed (CRLF)
setContentType	<ul style="list-style-type: none"> - 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

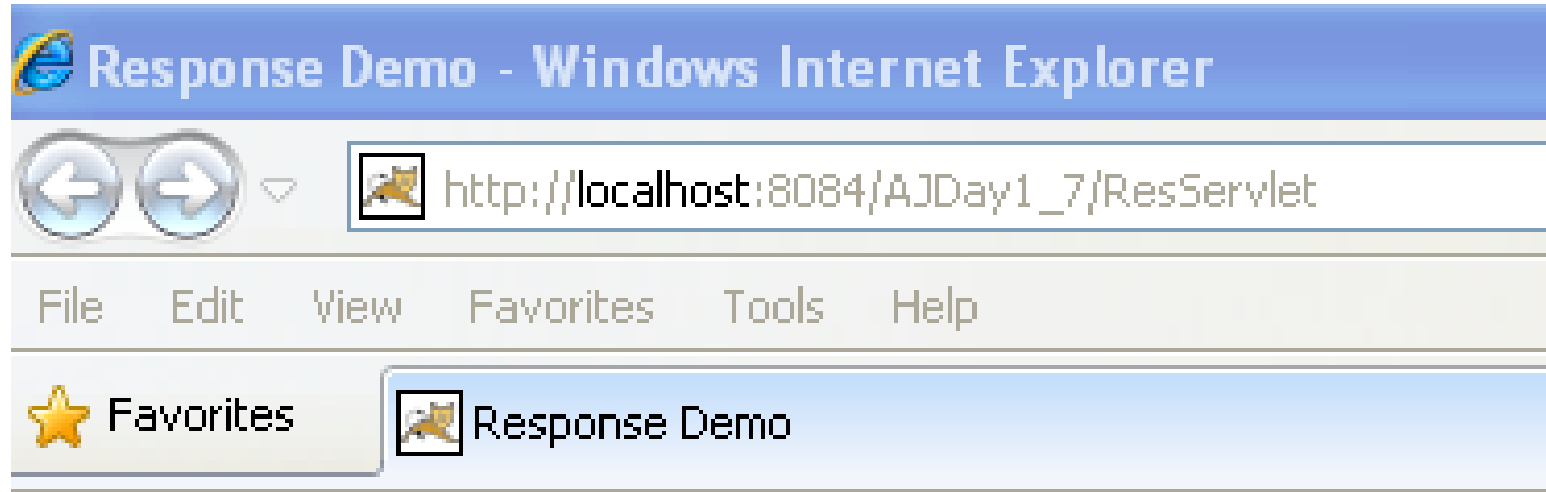
HttpServletResponse interface

- **Extends ServletResponse Interface**
- **Defines HttpServletResponse 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	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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: <code>response.sendRedirect("process.jsp");</code>

Appendix – The Servlet Model

HttpServletResponse interface - Example



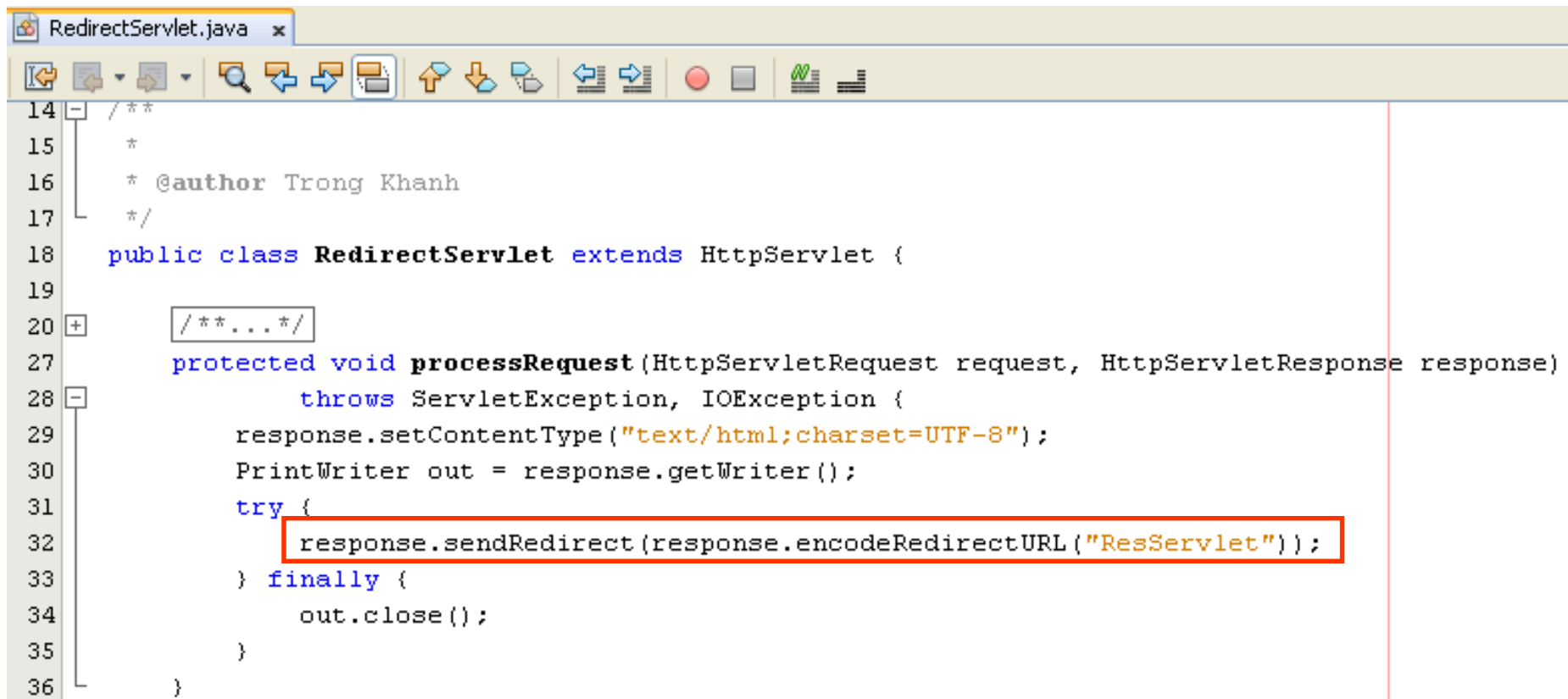
This is a Servlet Response

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

Appendix – The Servlet Model

HttpServletResponse interface - Example

- Using sendRedirect



```

14  /**
15   *
16   * @author Trong Khanh
17   */
18  public class RedirectServlet extends HttpServlet {
19
20      /**...*/
21
22      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
23          throws ServletException, IOException {
24          response.setContentType("text/html;charset=UTF-8");
25          PrintWriter out = response.getWriter();
26          try {
27              response.sendRedirect(response.encodeRedirectURL("ResServlet"));
28          } finally {
29              out.close();
30          }
31      }
32  }
    
```

Appendix – The Servlet Model

HttpServletResponse interface - Example

- ResServlet

```

ResServlet.java x
16  * @author Trong Khanh
17  */
18  public class ResServlet extends HttpServlet {
19
20      /** ... */
27  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
28      throws ServletException, IOException {
29      response.setContentType("text/html;charset=UTF-8");
30      PrintWriter out = response.getWriter();
31      try {
32          out.println("<html>");
33          out.println("<head>");
34          out.println("<title>Response Demo</title>");
35          out.println("</head>");
36          out.println("<body>");
37          out.println("<h1>This is a Servlet Response</h1>");
38
39          out.println("Content Type: " + response.getContentType() + "<br/>");
40
41          out.println("</body>");
42          out.println("</html>");
43      } finally {
44          out.close();
45      }
46  }
  
```


The Servlet Model

HttpServlet class

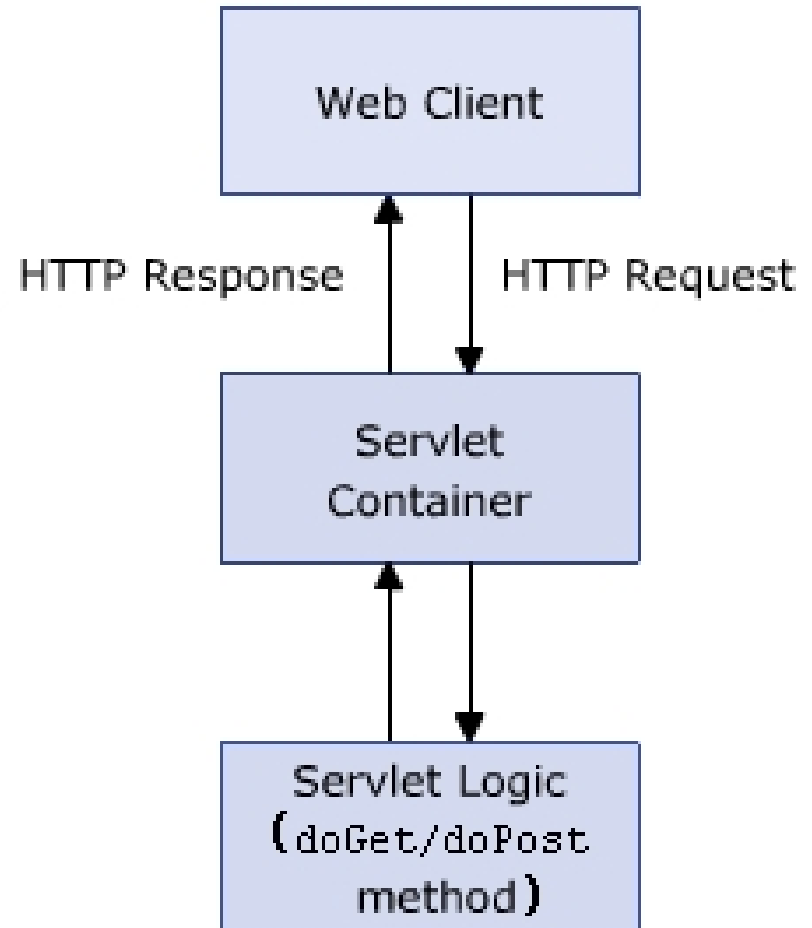
- The protocol **defines** a set of **text-based request messages** called HTTP ‘methods’ **implemented in *HttpServlet* class**
- Provides **an abstract class** to create an **HTTP Servlet**
- **Extends the *GenericServlet* class**
- A subclass of *HttpServlet* class **must override at least one** of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

Methods	Descriptions
doGet	<ul style="list-style-type: none">- 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	<ul style="list-style-type: none">- 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

```
login.html x
Source History
1 <!DOCTYPE html>
2 ...5 lines
7 <html>
8   <head>
9     <title>Login</title>
10    <meta charset="UTF-8">
11    <meta name="viewport" content="width=device-width, initial-scale=1.0">
12  </head>
13  <body>
14    <h1>Login Page</h1>
15
16    <form action="SE1162Servlet" method="POST">
17      Username <input type="text" name="txtUsername" value="" /><br/>
18      Password <input type="password" name="txtPassword" value="" /><br/>
19      <input type="submit" value="Login" name="btAction" />
20      <input type="reset" value="Reset" />
21    </form>
```

Appendix – Build The Simple Web Invalid Page

```

invalid.html x
Source History
1 <!DOCTYPE html>
2 ...5 lines
7 <html>
8   <head>
9     <title>Invalid</title>
10    <meta charset="UTF-8">
11    <meta name="viewport" content="width=device-width, initial-scale=1.0">
12  </head>
13  <body>
14    <h1>
15      <font color="red">
16        Invalid username or password!!!
17      </font>
18    </h1>
19
20    <a href="login.html">Click here to try again</a><br/>
  
```

Appendix – Build The Simple Web Search Page

```

search.html x
Source History
1 <!DOCTYPE html>
2 ...5 lines
7 <html>
8   <head>
9     <title>Search</title>
10    <meta charset="UTF-8">
11    <meta name="viewport" content="width=device-width, initial-scale=1.0">
12  </head>
13  <body>
14    <h1>Search Page</h1>
15    <form action="SE1162Servlet">
16      Search Value <input type="text" name="txtSearchValue" value="" /><br/>
17      <input type="submit" value="Search" name="btAction" />
18    </form>
19  </body>
20 </html>
21
  
```

Appendix – Build The Simple Web Servlet

```

SE1162Servlet.java x
Source History
23  * @author kieukhanh
24  */
25  public class SE1162Servlet extends HttpServlet {
26      private final String searchPage = "search.html";
27      private final String invalidPage = "invalid.html";
28      /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines *
29
30      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
31          throws ServletException, IOException {
32          response.setContentType("text/html;charset=UTF-8");
33          PrintWriter out = response.getWriter();
34          try {
35              String button = request.getParameter("btAction");
36              String url = invalidPage;
37              if (button.equals("Login")) {
38                  String username = request.getParameter("txtUsername");
39                  String password = request.getParameter("txtPassword");
40
41                  RegistrationDAO dao = new RegistrationDAO();
42                  boolean result = dao.checkLogin(username, password);
43
44                  if (result) {
45                      url = searchPage;
46                  }
47              }
48              response.sendRedirect(url);
49          } catch (NamingException ex) {
50              ex.printStackTrace();
51          } catch (SQLException ex) {
52              ex.printStackTrace();
53          } finally {
54              out.close();
55          }
56      }
57  }
58
59
60
61

```

Appendix – Build The Simple Web DAO

RegistrationDAO.java

```

20  * @author kieukhanh
21  */
22  public class RegistrationDAO implements Serializable {
23      public boolean checkLogin(String username, String password)
24          throws SQLException, NamingException {
25          Connection con = null;
26          PreparedStatement stm = null;
27          ResultSet rs = null;
28          try {
29              con = DBUtils.makeConnection();
30              if (con != null) {
31                  String sql = "Select * From Registration Where username = ? And password = ?";
32
33                  stm = con.prepareStatement(sql);
34                  stm.setString(1, username);
35                  stm.setString(2, password);
36
37                  rs = stm.executeQuery();
38                  if (rs.next()) {
39                      return true;
40                  }
41              }
42          } finally {
43              if (rs != null) {
44                  rs.close();
45              }
46              if (stm != null) {
47                  stm.close();
48              }
49              if (con != null) {
50                  con.close();
51              }
52          }
53          con.close();
54      }
55  }
56
57  return false;
58  }

```