The Servlet Model

HTTP Methods
Form Parameters
Requests
Responses
Servlet Life Cycle

Objectives

- **HTML Introduction**
  - What is HTML?
  - HTML Tags
  - Web Browsers
- **The Servlet Model**
  - HTML Methods (GET, POST)
  - Form Parameters
  - Requests
  - Responses
  - Servlet Life Cycle

HTML Introduction

What is HTML?

- **HTML** is a **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

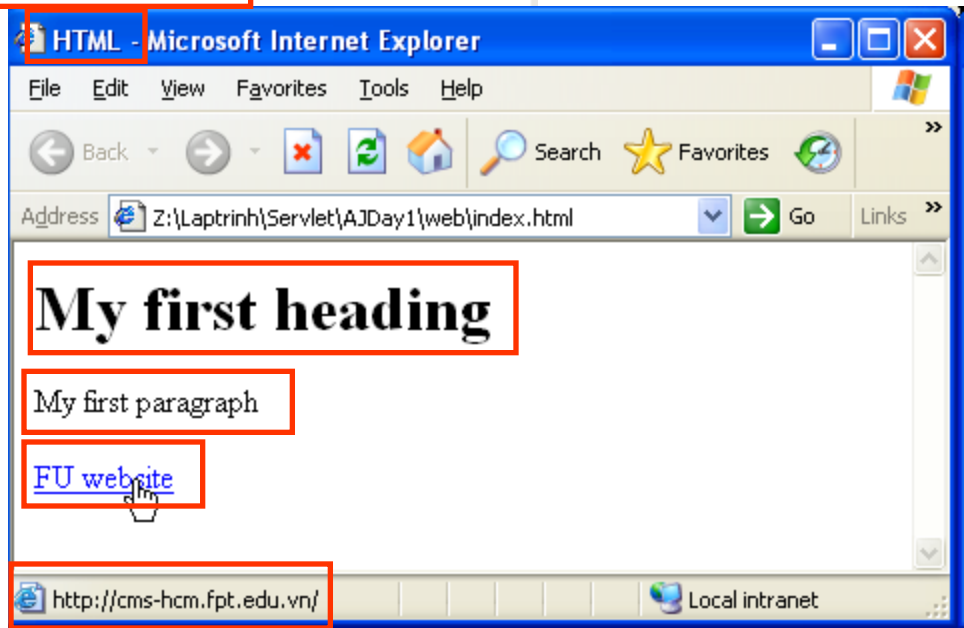
- HTML markup tags are usually called **HTML tags**
  - HTML tags are keywords surrounded by **angle brackets, that begin "<"** and **finish with ">",** like <html>
  - HTML tags normally **come in pairs** like and
    - The first tag in a pair is the **start tag,** the second tag is the **end tag**
    - 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

index.html

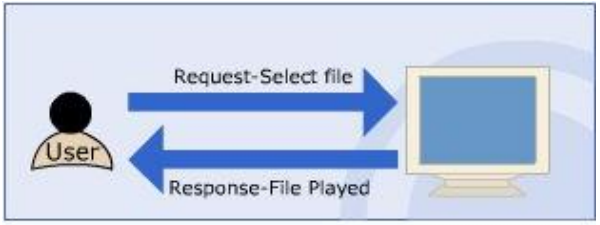
```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3    <head>
4      <title>HTML</title>
5      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6    </head>
7    <body>
8      <h1>My first heading</h1>
9      <p>My first paragraph</p>
10     <a href="http://cms-hcm.fpt.edu.vn">FU website</a>
11   </body>
12 </html>
```



The Servlet Model

Applications

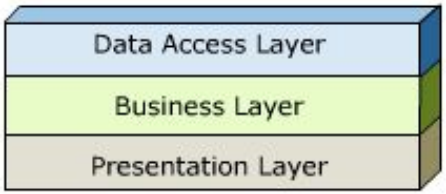
- A **collection of program** is **designed** to **perform** a **particular task** (different purposes)
- **Classification** (based on running and accessibility)



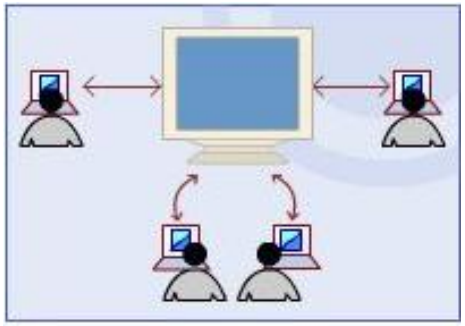
Desktop Application

Local machine

Single user



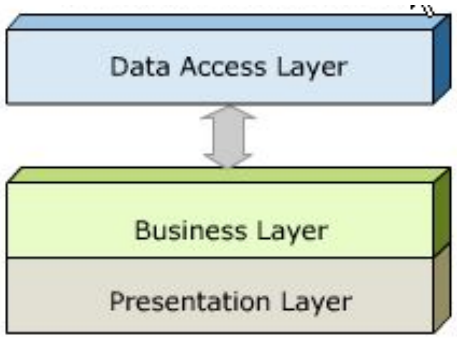
One-tier Architecture



Network Application

LAN, WAN, or MAN

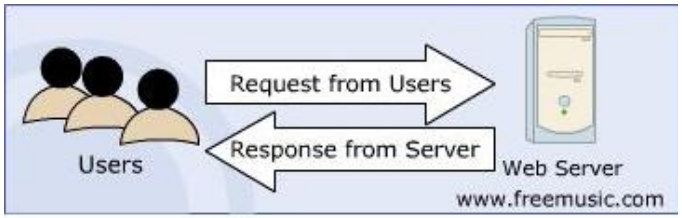
Muli-users in particular network only



Two-tier Architecture

The Servlet Model

Applications



Web Application

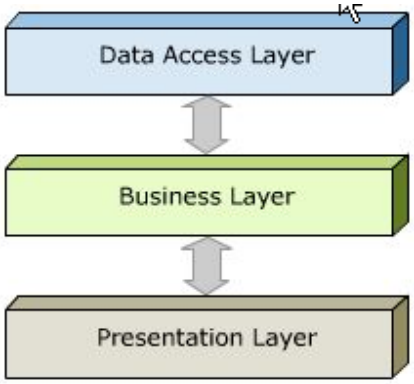
Web Server

Multi-users having administrator or equivalent privileges

Browsing with Web Browser

N – tiers Architecture

- Subdivided to functioning
- Presentation is GUI
- Reducing the number location implementing the logic

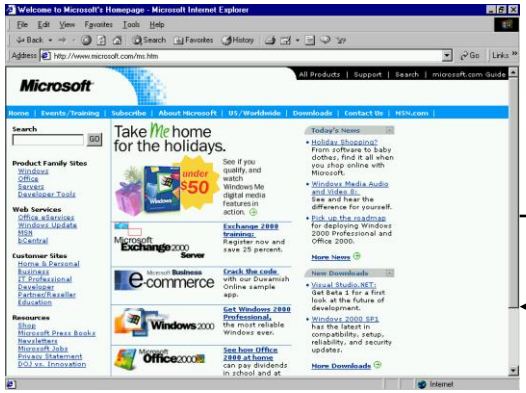


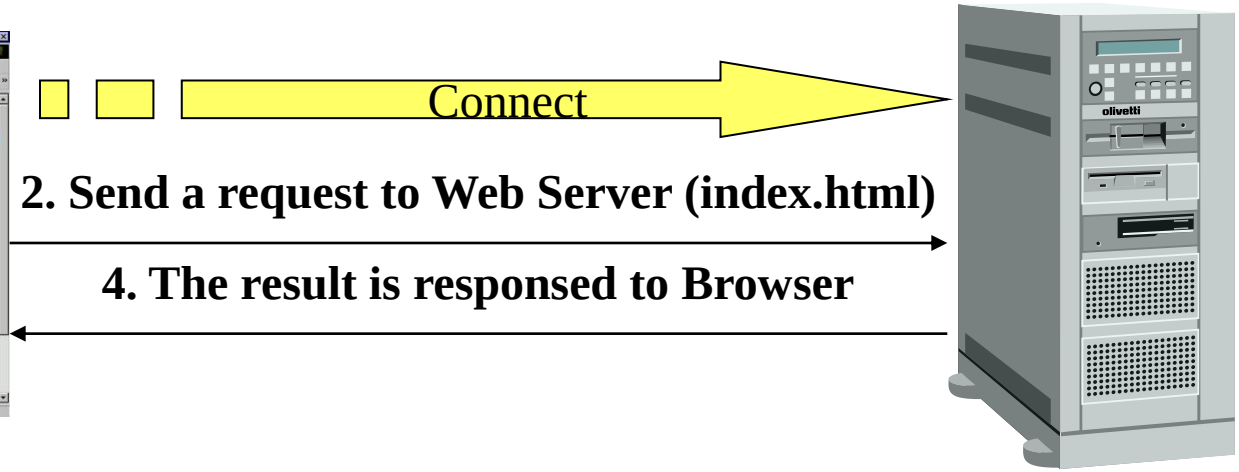
Three-tier Architecture

The Servlet Model

HTTP Protocols

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





http://microsoft.com/index.html

192.168.54.3:80

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

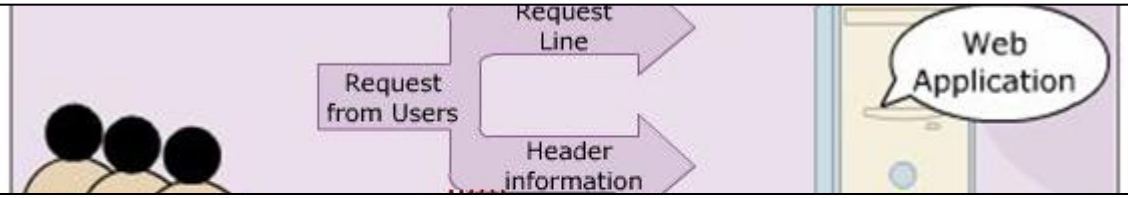
5. Web Browser views the result which contains a markup language

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

The Servlet Model

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



- Return the User-Agent (the **browser**) along **with** the **Accept header**

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

- **Ex**: User-Agent: Mozilla/4.0 (compatible: MSIE 4.0 : Windows 95)
Accept : image/gif, image/jpeg, text/*, */*

The Servlet Model

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
```

The Servlet Model

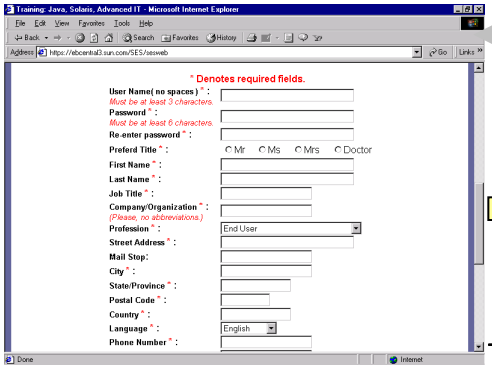
HTTP Requests – Example

```
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

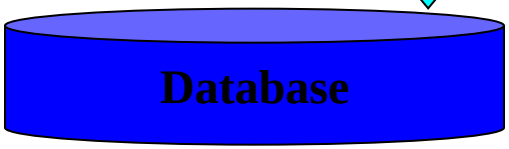


1. Form's information is sent to Web Server using request parameter.

4. The result of processing is responsed

2. Server process requested client (server script – Server Side), connect DB ...

3. Connect



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 **length of query string**, that is introduced by the question mark "?", is **restricted** 240 to 255
- 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**> whose **method** is set to **GET**

The Servlet Model

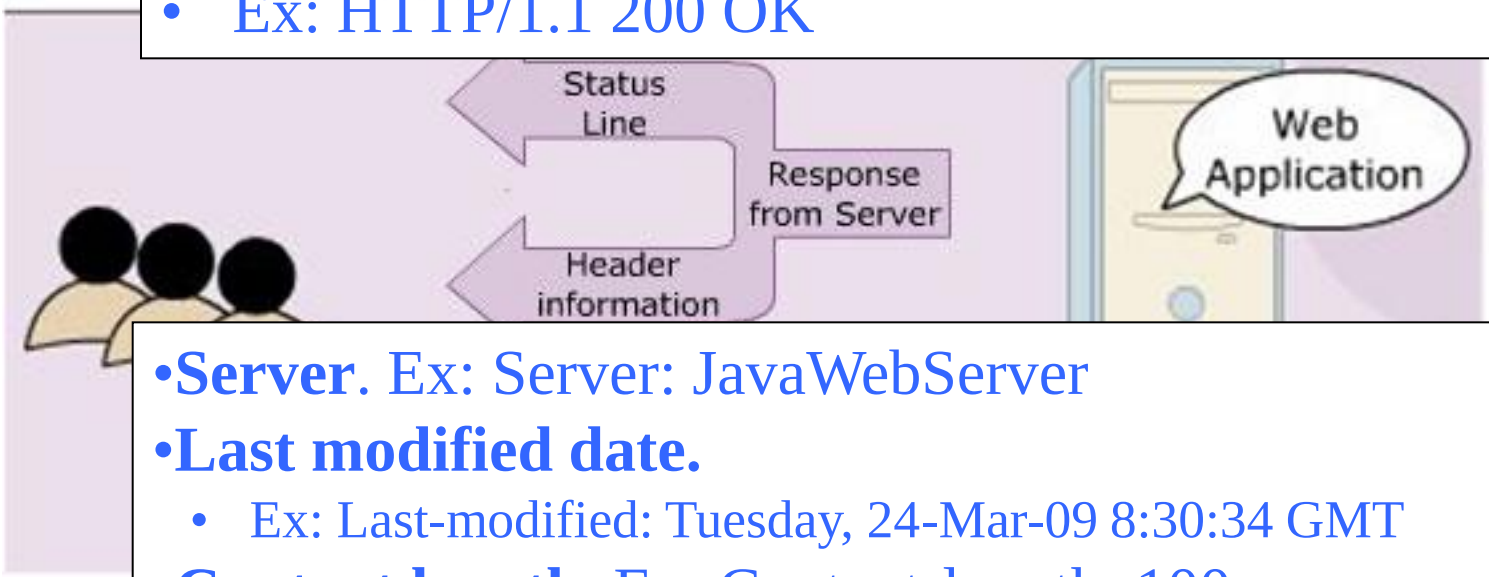
HTTP Methods

- **GET** – **retrieves** the resource identified by the request URL
- **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
- **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** – **stores** a **resource** under the request URL.
- **DELETE** – **removes** the **resource** identified by the request URL.
- **TRACE** – **returns** the **header fields** sent with the **TRACE request**.
- **Idempotency** and **Safety**
  - GET, TRACE, OPTIONS, and HEAD

The Servlet Model

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

The Servlet Model

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
```

The Servlet Model

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
```

The Servlet Model

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

The Servlet Model

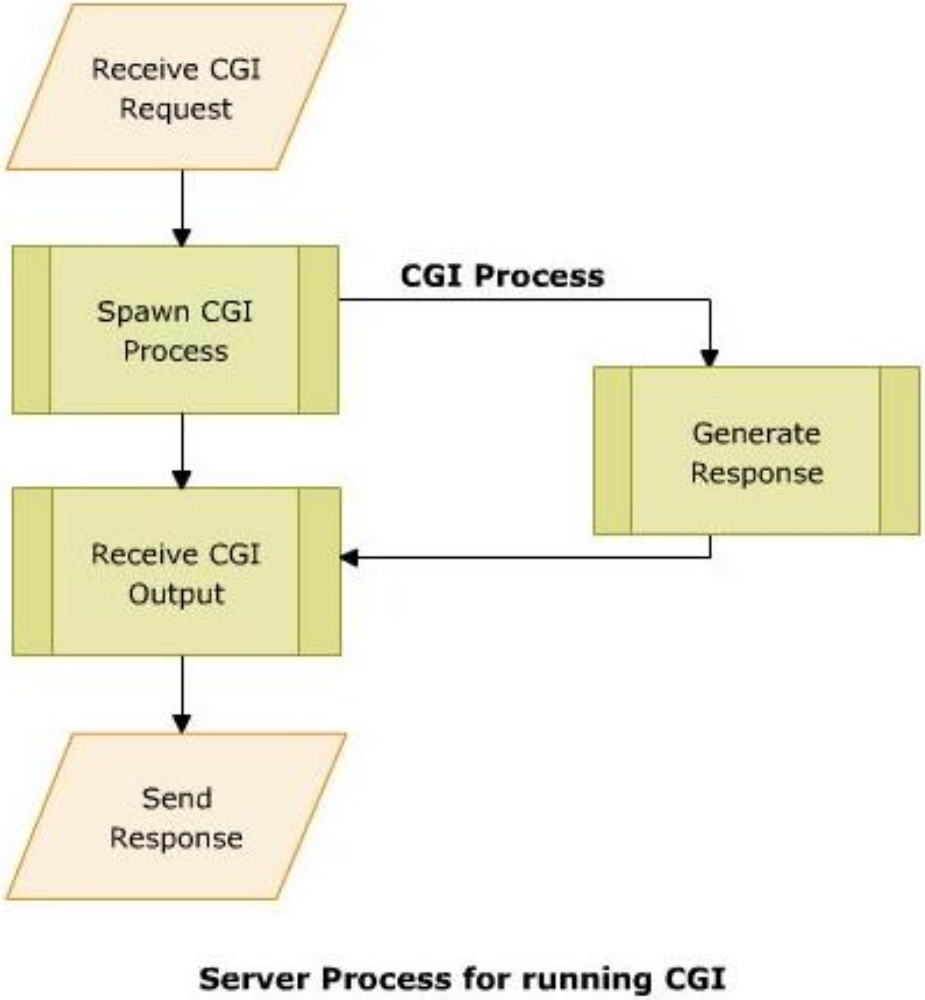
Some commonly Status codes

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
…	…	-…

The Servlet Model

Common Gateway Interface (CGI)

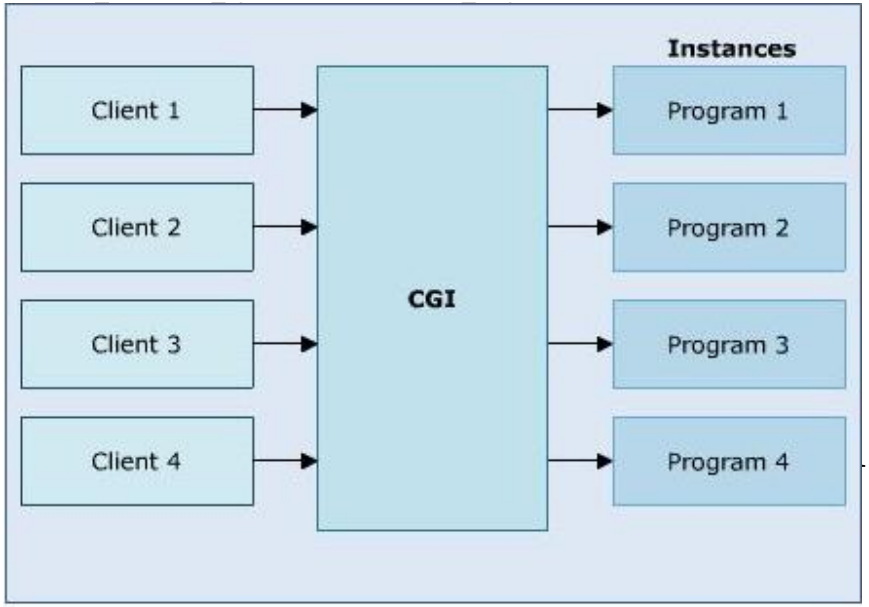
- A **small program (*.exe)** is **written** in **languages** such as **C/C++, Perl**, ... for the gateway programs.
- Used in complex applications, such as **Web pages**
- A set of standards followed to **interface applications form client side** to a Web Server
- Enables the Web server to send information 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**.



The Servlet Model

Common Gateway Interface (CGI)

- Disadvantages
  - **Reduced efficiency**
  - for graphical or highly
  - **memory consumed**
  - s difficult



CGI process

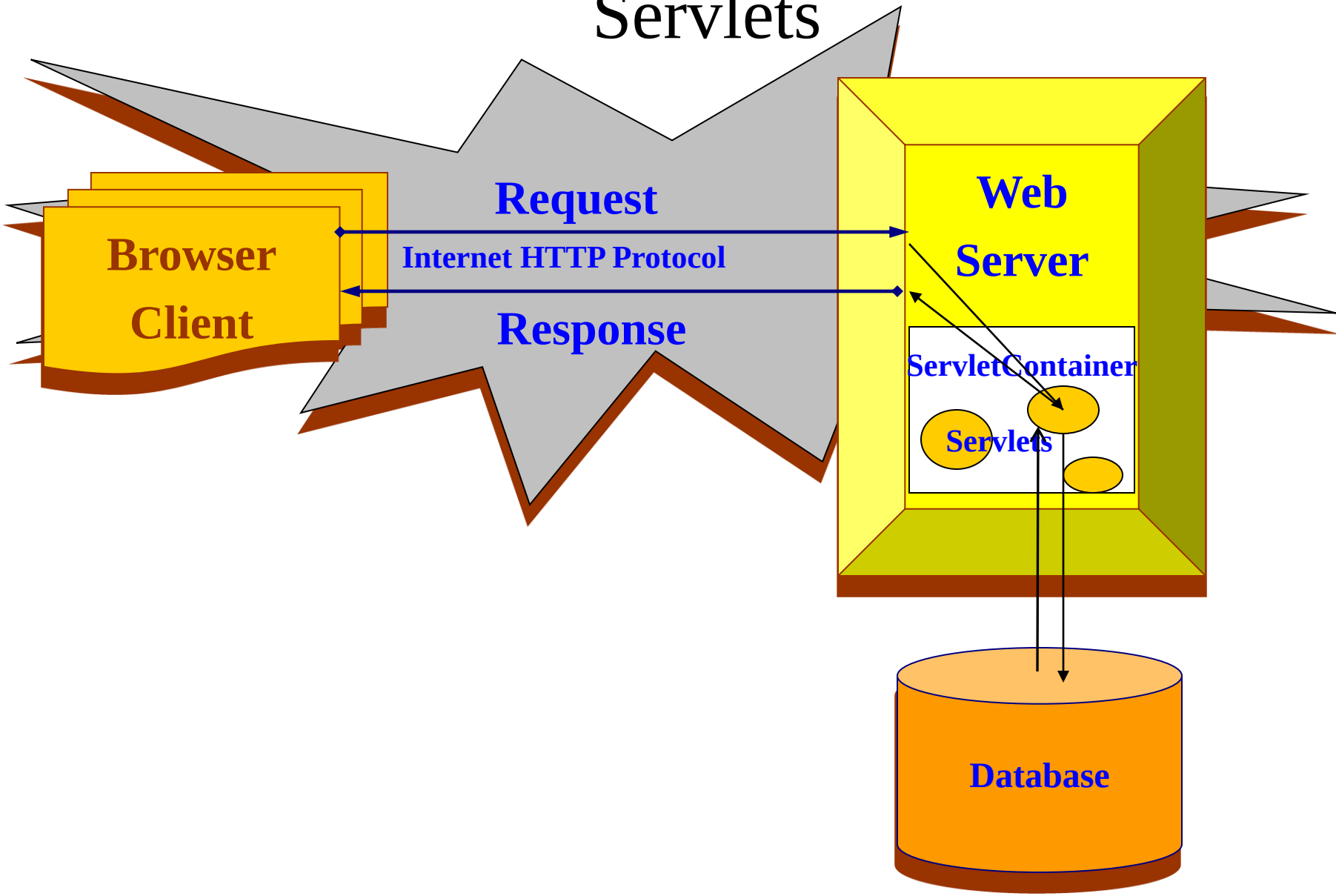
The Servlet Model

Servlets

- Are **small Java programs** that **run** on a **Web server** and help to **build dynamic Web pages.**
- Servlets **receive** and **respond** to requests from Web clients, usually across HTTP.
- Java Servlet technology was created as a **portable way** to **provide dynamic, user-oriented content.**
- A server side scripting is **not requirement** of **reloading** the **Servlet compiler each time a request** is received from client
- Using **multi threading** (**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

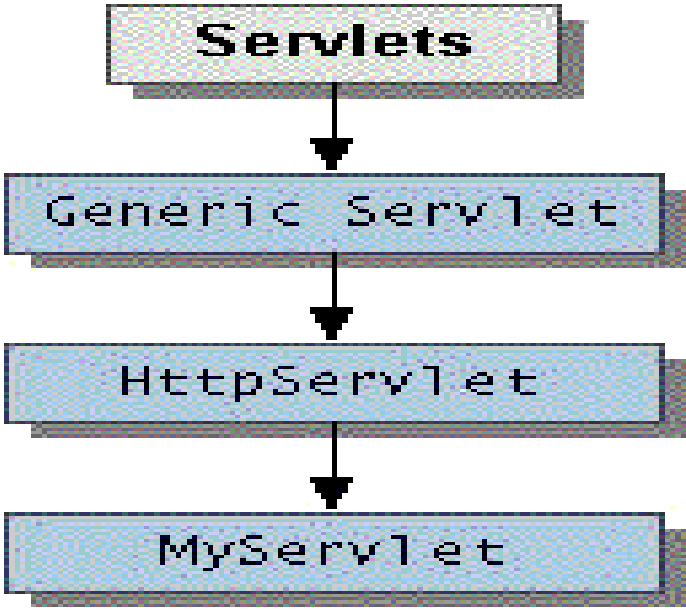
Servlets



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

Form Parameters

- HTML Forms
  - 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

The Servlet Model

Form Parameters

- HTML Forms – **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**

The Servlet Model

Form Parameters

- 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

The Servlet Model

Form Parameters – Examples

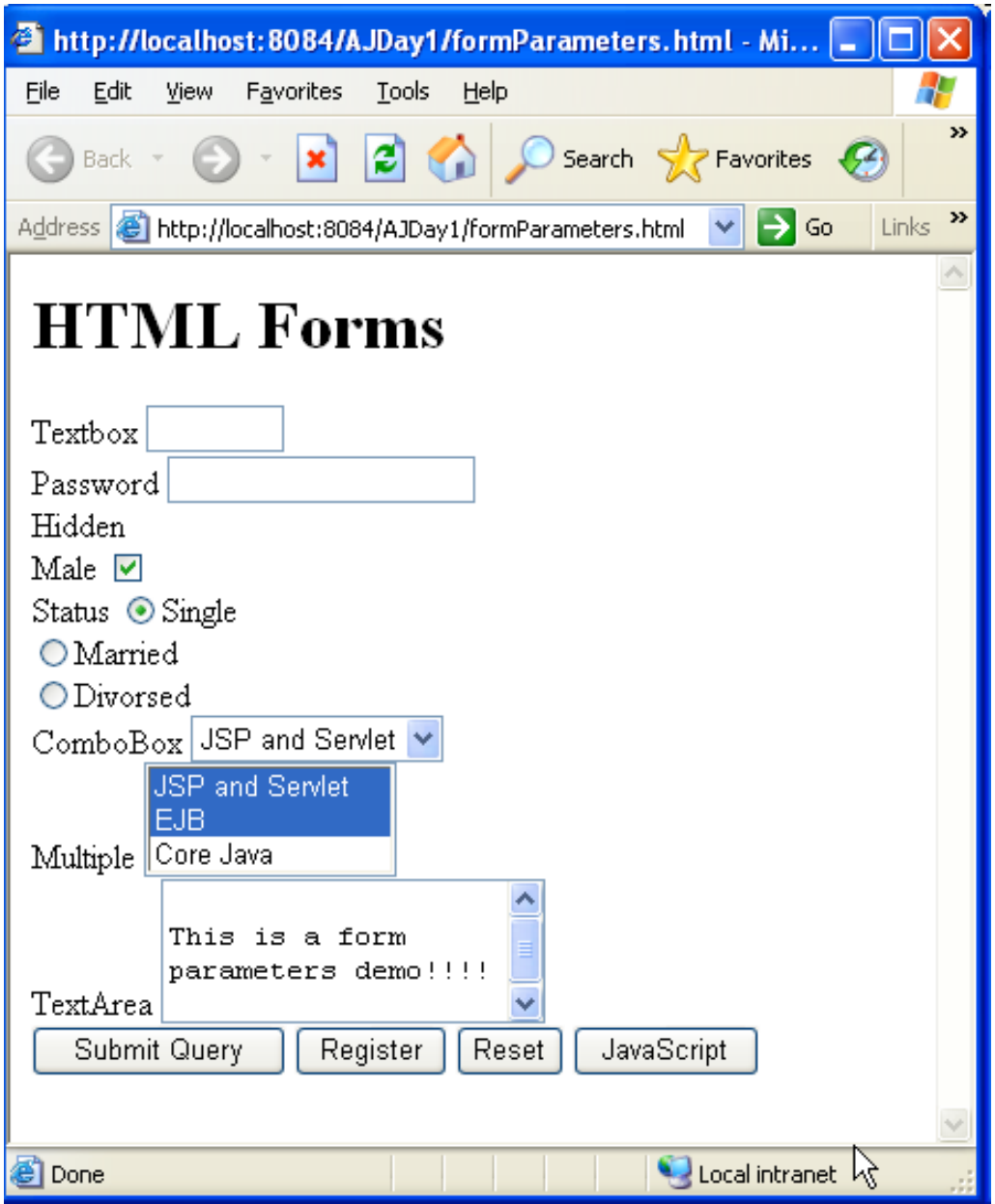
formParameters.html

html

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

The Servlet Model

Form Parameters – Examples



Web Applications

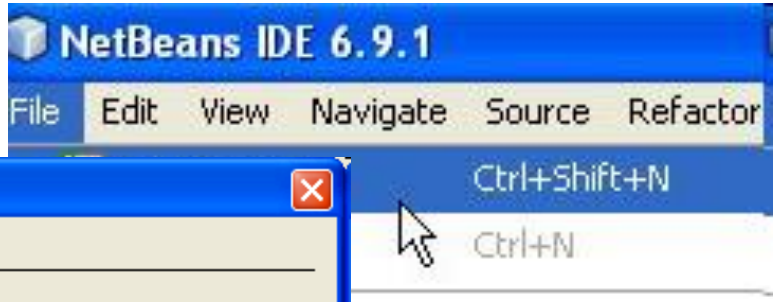
Web Application Development Process

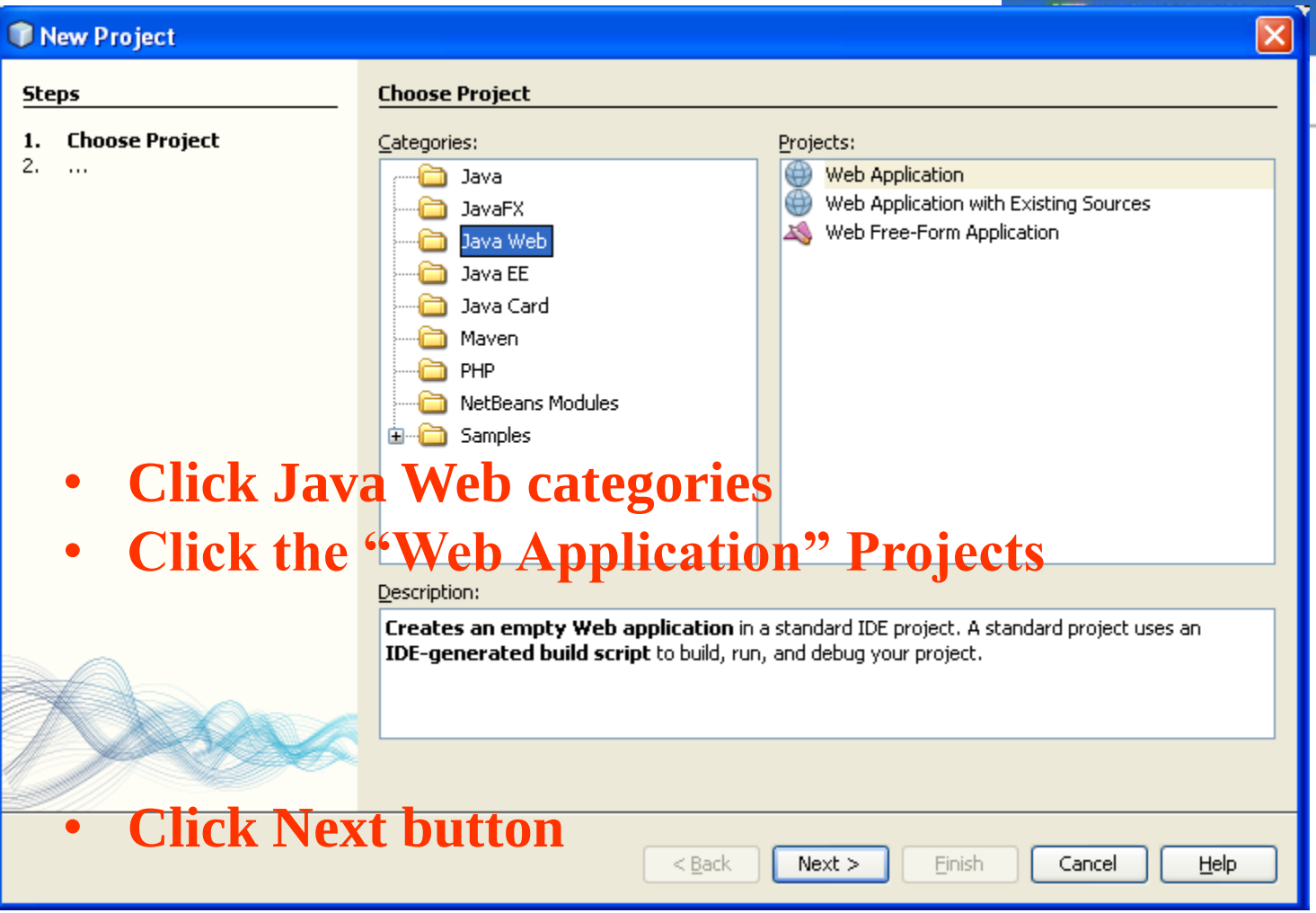
- **Requirement tools: NetBeans 6.9.1**
- **Step 1:** Creating a Web application project
- **Step 2:** Creating the Servlets
- **Step 3:** Writing the code for Servlet & Compile
- **Step 4:** Building the Web application project
- **Step 5:** Deploying to a Web Server
- **Step 6:** Executing the application

Web Applications

Web Application Development Process

- **Step 1:** Creating a Web App project



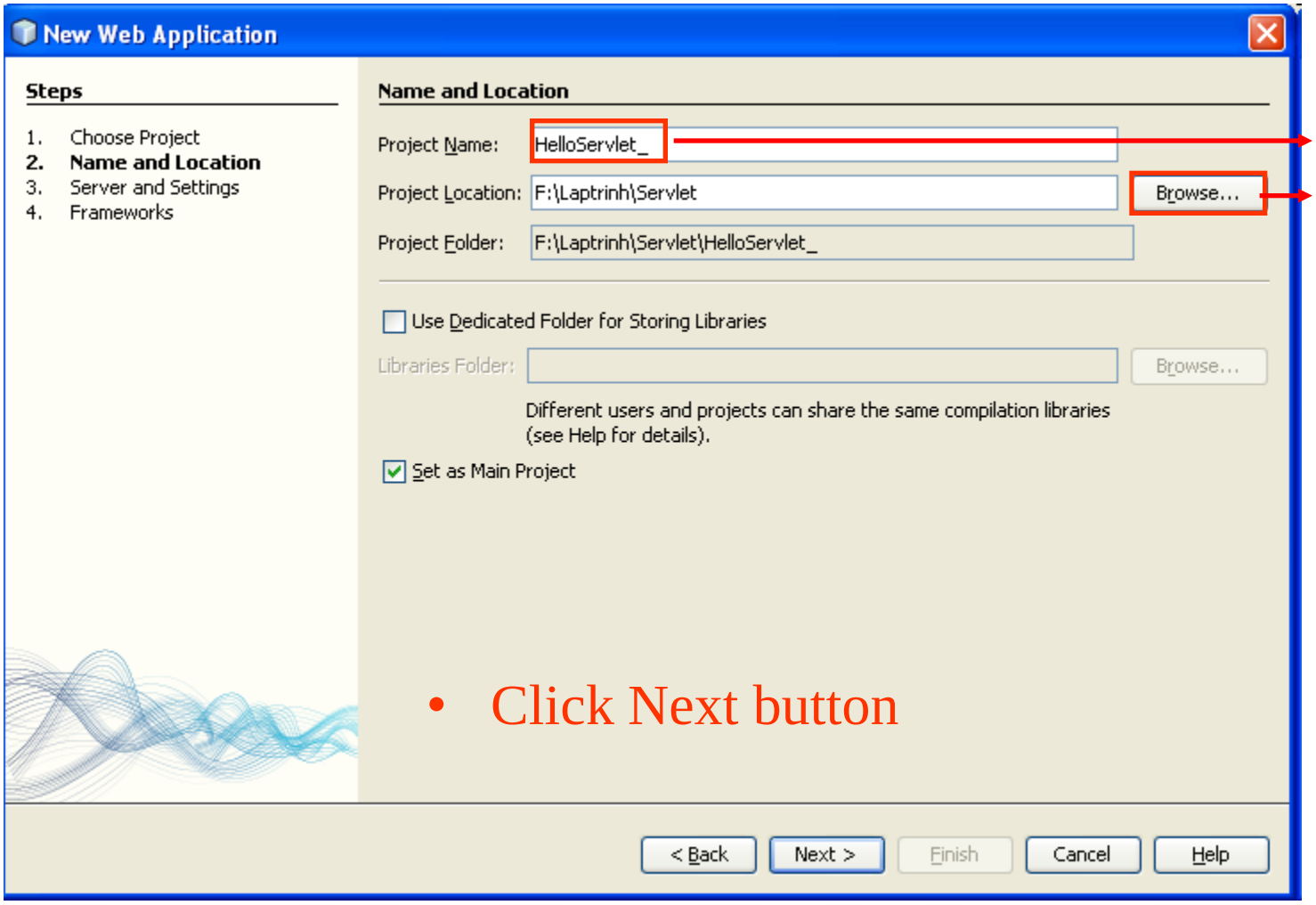


- **Click Java Web categories**
- **Click the "Web Application" Projects**
- **Click Next button**

Web Applications

Web Application Development Process

- **Step 1:** Creating a Web App project



Fill your project name

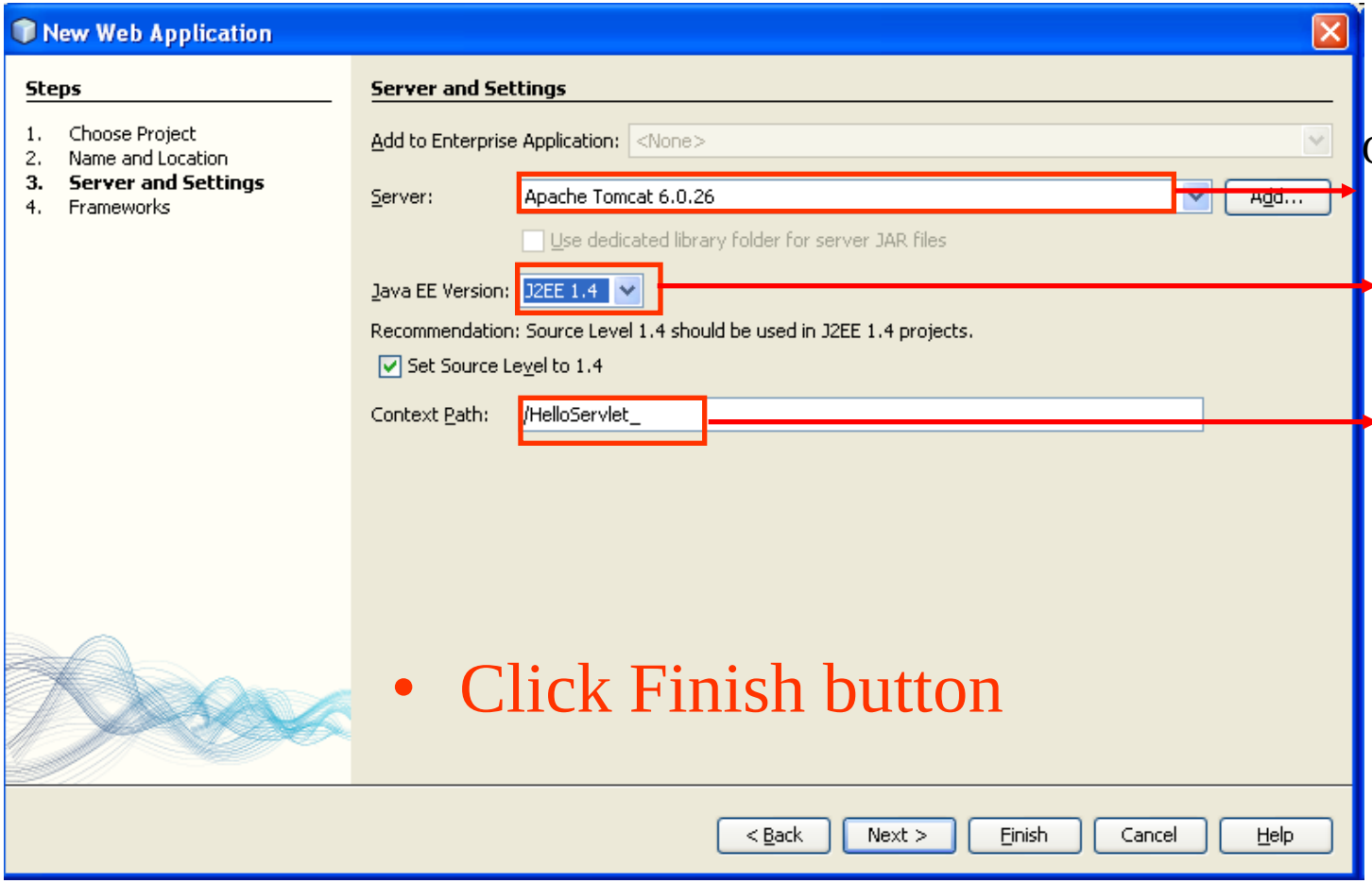
Browser your location where store the project

- Click Next button

Web Applications

Web Application Development Process

- **Step 1:** Creating a Web App project



Choose deployed server

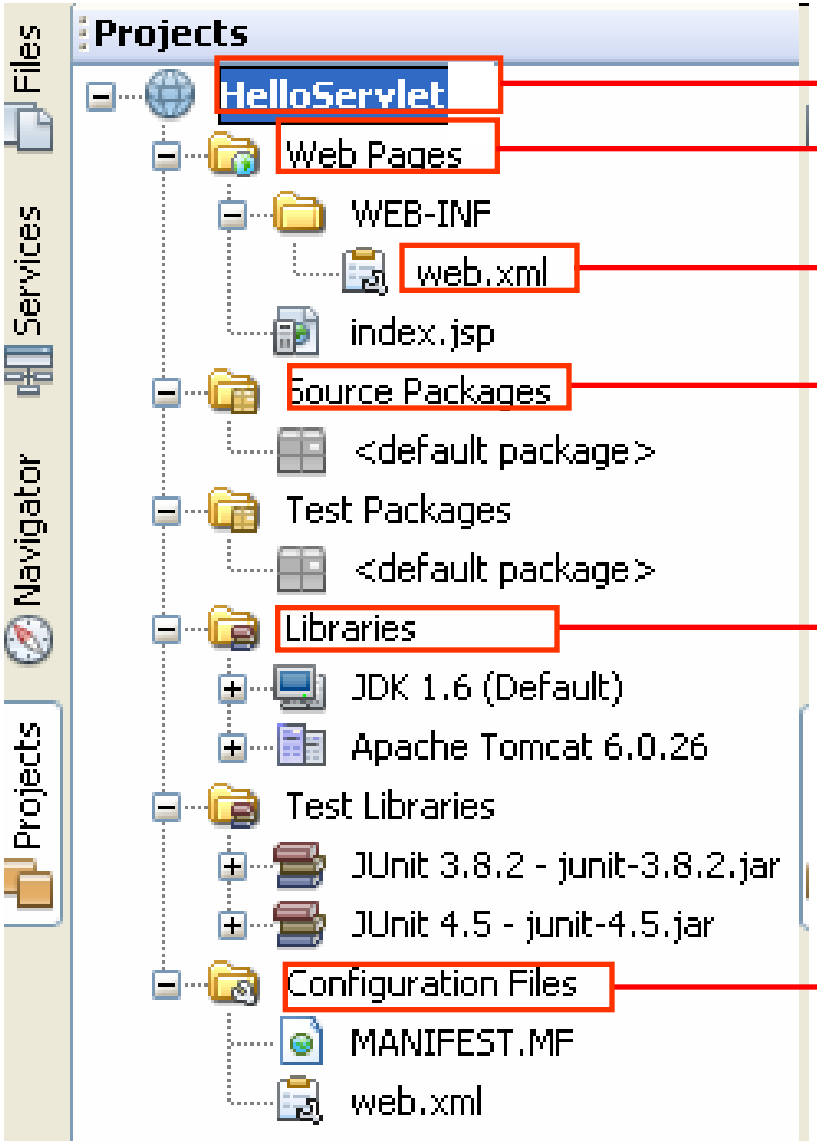
Choose J2EE 1.4

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

- Click Finish button

Web Applications

Web Application Development Process



- **Step 1:** Creating a Web App project

Project name

Web Directory

Web deployment descriptor

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

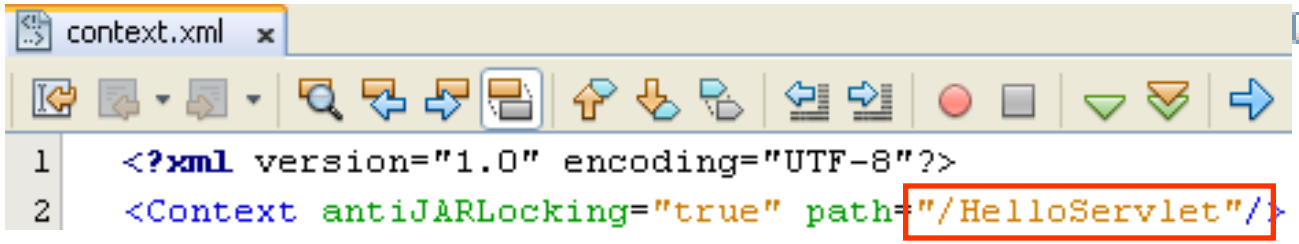
Support library directory, containing **jar file**. When project is built, package in **lib directory**

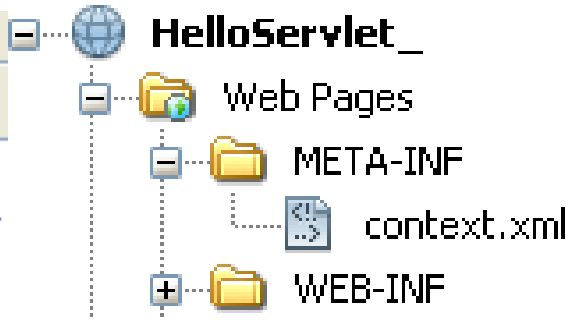
Configuration directory related define for Web App

Web Applications

Add the META-INF/context.xml to project

- **Step 1:** Creating a Web App 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**)

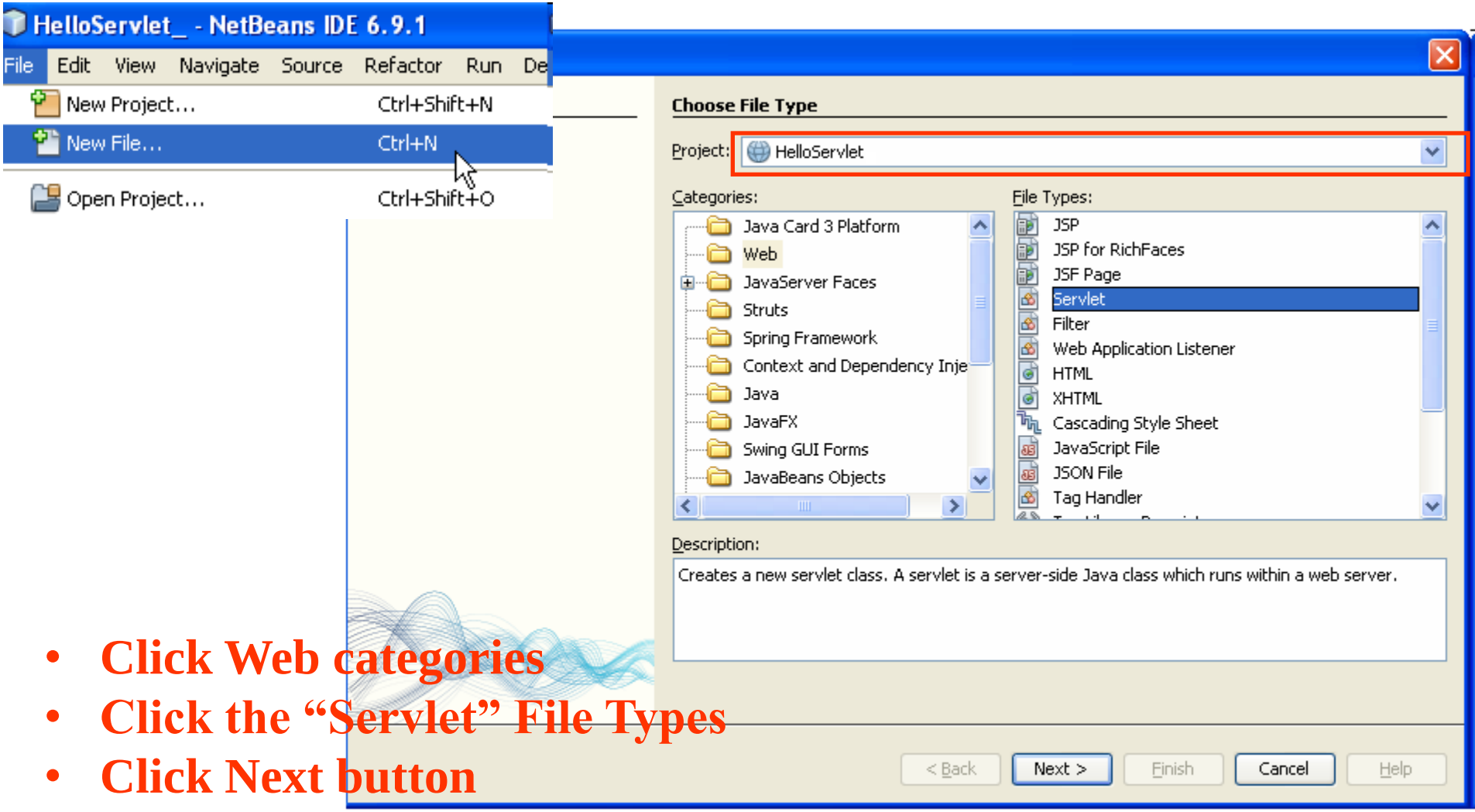




Web Applications

Web Application Development Process

- **Step 2:** Creating a Servlet

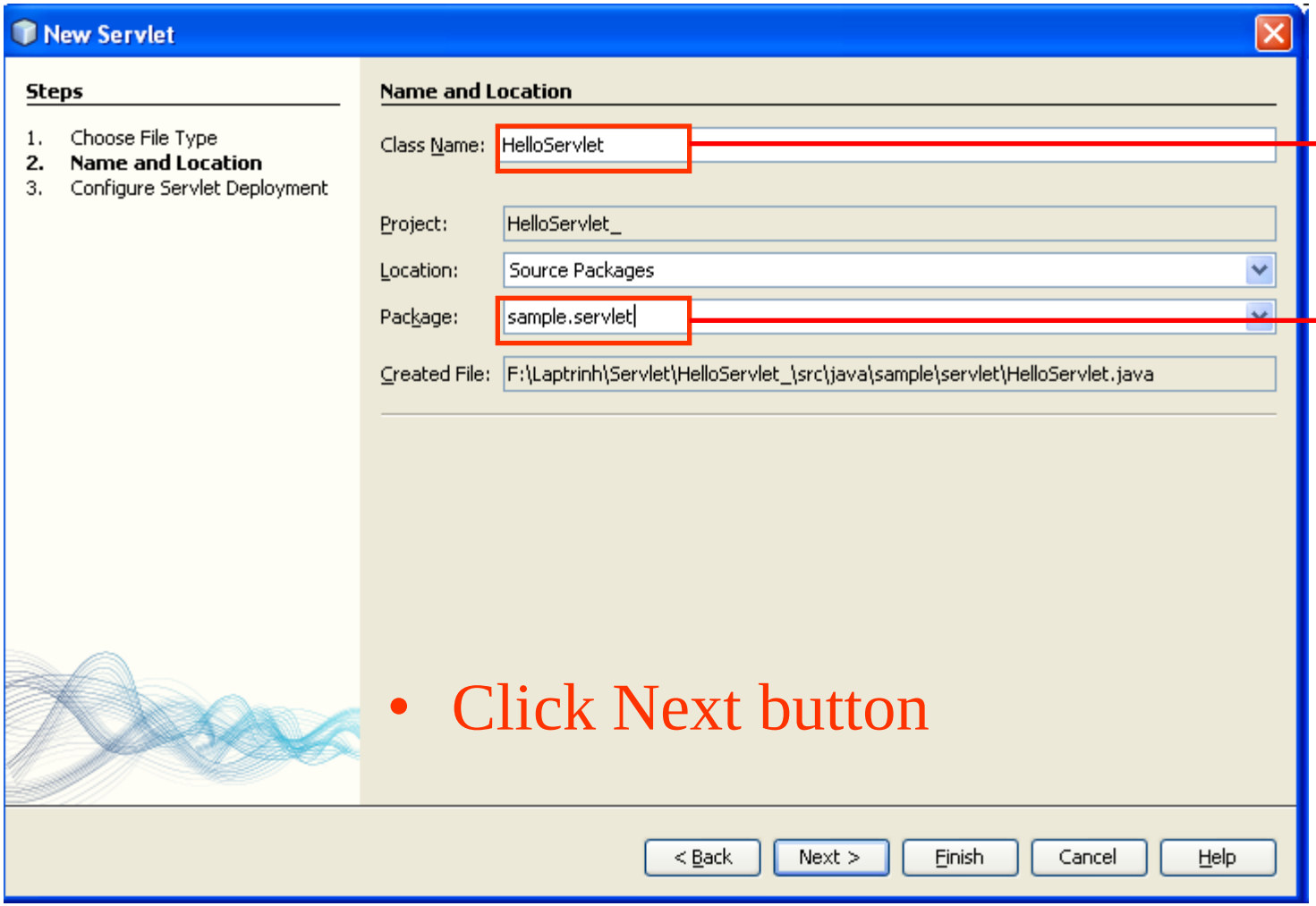


- **Click Web categories**
- **Click the "Servlet" File Types**
- **Click Next button**

Web Applications

Web Application Development Process

- **Step 2:** Creating a Servlet



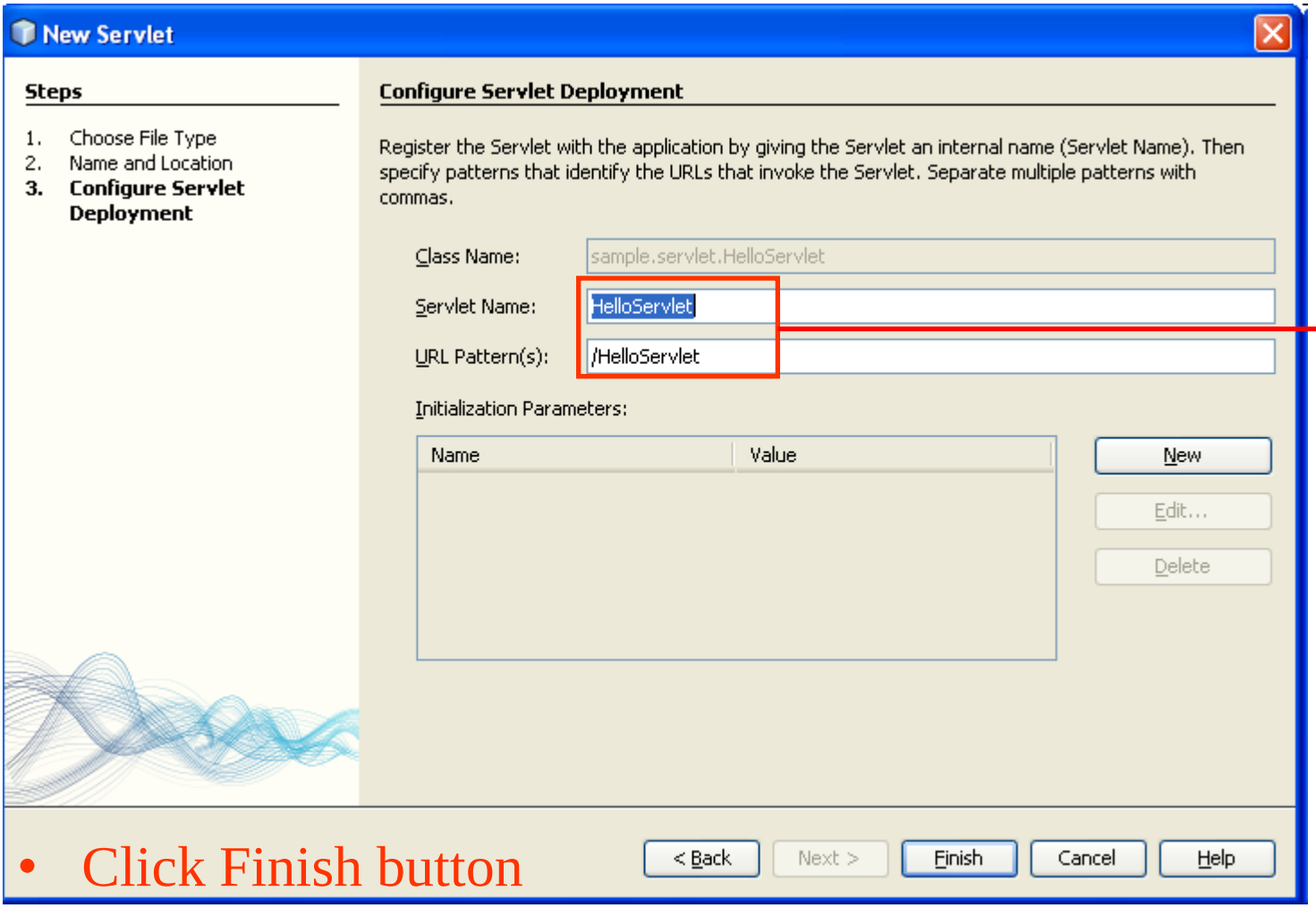
Fill your servlet name

Fill or choose package name

- Click Next button

Web Applications

Web Application Development Process



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

Web Applications

Web Application Development Process

- **Step 3:** Writing the Code and Compile

HelloServlet.java

```
28      protected void processRequest(HttpServletRequest request,
29              HttpServletResponse response)
30              throws ServletException, IOException {
31          response.setContentType("text/html;charset=UTF-8");
32          PrintWriter out = response.getWriter();
33          try {
34              out.println("<html>");
35              out.println("<head>");
36              out.println("<title>Hello</title>");
37              out.println("</head>");
38              out.println("<body>");
39              out.println("<h1>Hello the Servlet World!!!</h1>");
40              out.println("</body>");
41              out.println("</html>");
42
43          } finally {
44              out.close();
45          }
46      }
```

Web Applications

Web Application Development Process

- **Step 3:** Writing the Code and Compile

web.xml

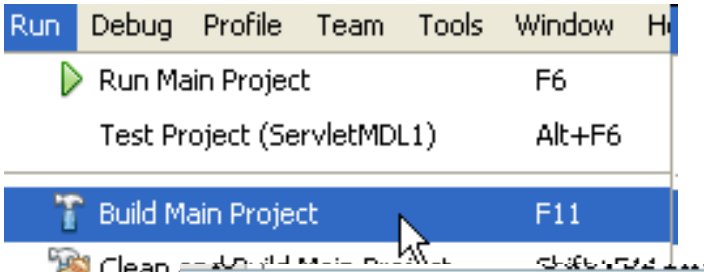
General | Servlets | Filters | Pages | References | Security | XML

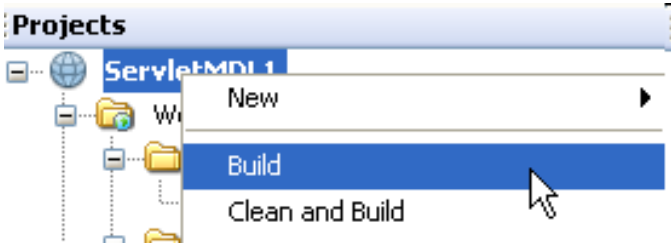
```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <servlet>
        <servlet-name>HelloServlet</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>HelloServlet</welcome-file>
    </welcome-file-list>
</web-app>
```

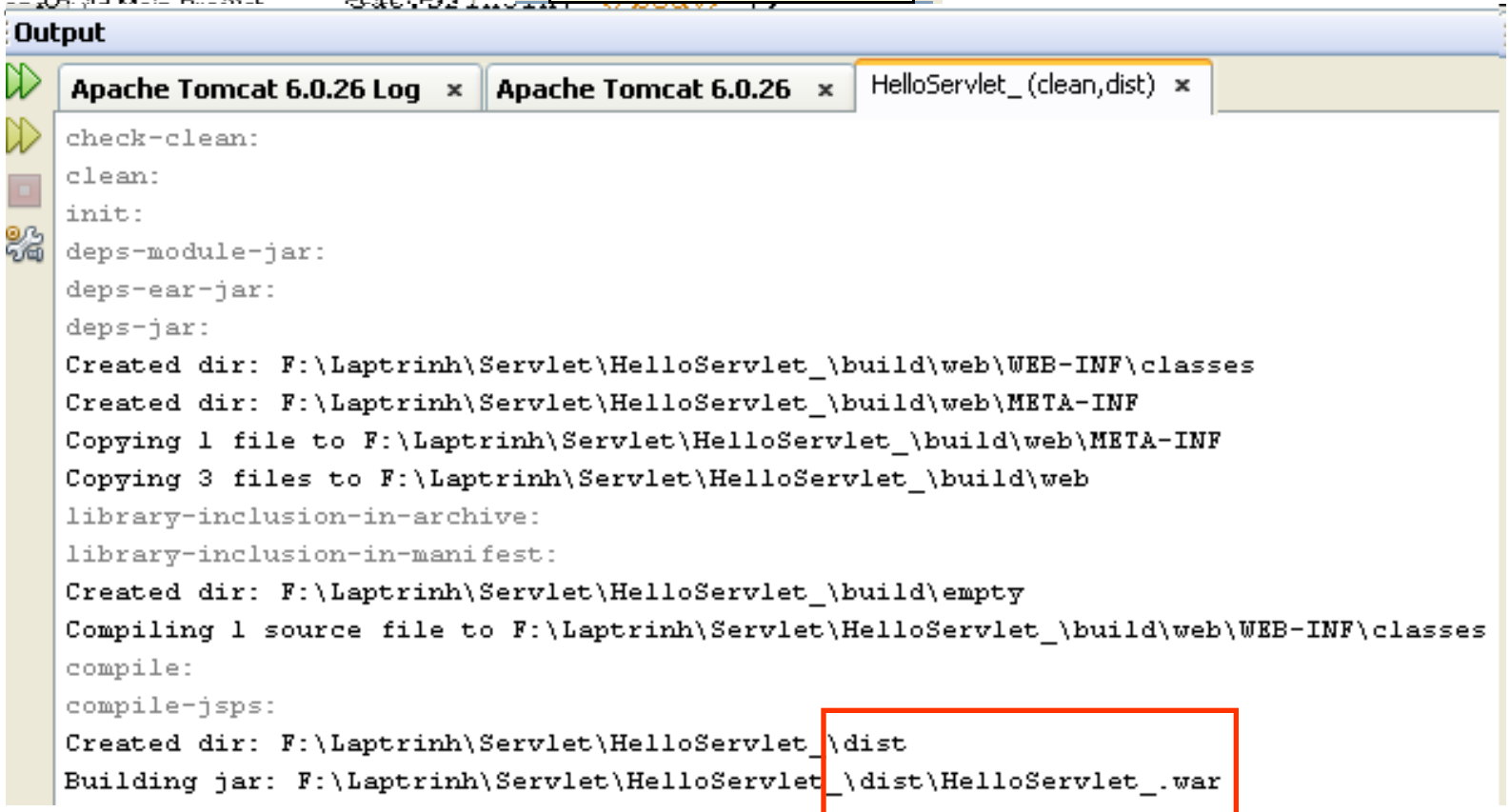
Web Applications

Web Application Development Process

- **Step 3:** Writing the Code and Compile







Web Applications

Web Application Development Process

- **Step 3:** Writing the Code and Compile

g:\Laptrinh\Servlet\HelloServlet\dist*.*

Name	Ext
[..]	
HelloServlet	war

g:\Laptrinh\Servlet\HelloServlet\dist\HelloServlet.war*.*

Name	Ext	Size
[..]		<DIR>
[META-INF]		<DIR>
[WEB-INF]		<DIR>
index	jsp	996

g:\Laptrinh\Servlet\HelloServlet\build\web*.*

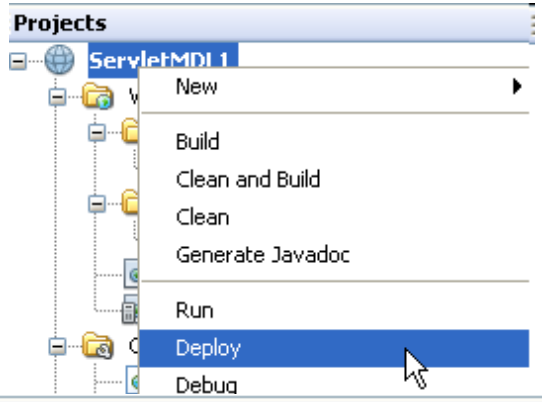
Name	Ext
[..]	
[META-INF]	
[WEB-INF]	
index	jsp

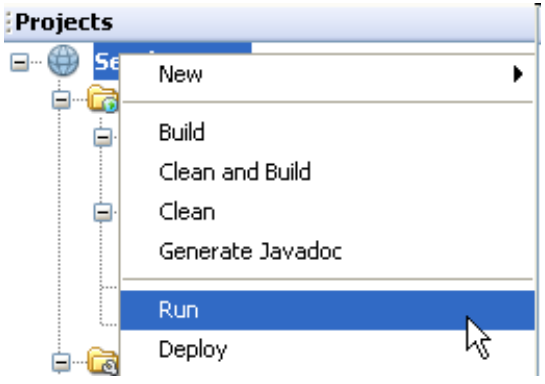
- Package War file with command prompt
  - **jar –cvf** fileName.war directoryOrFile (using blank to separate)
  - **Ex**: jar –cvf HelloServlet.war *.jsp WEB-INF/*

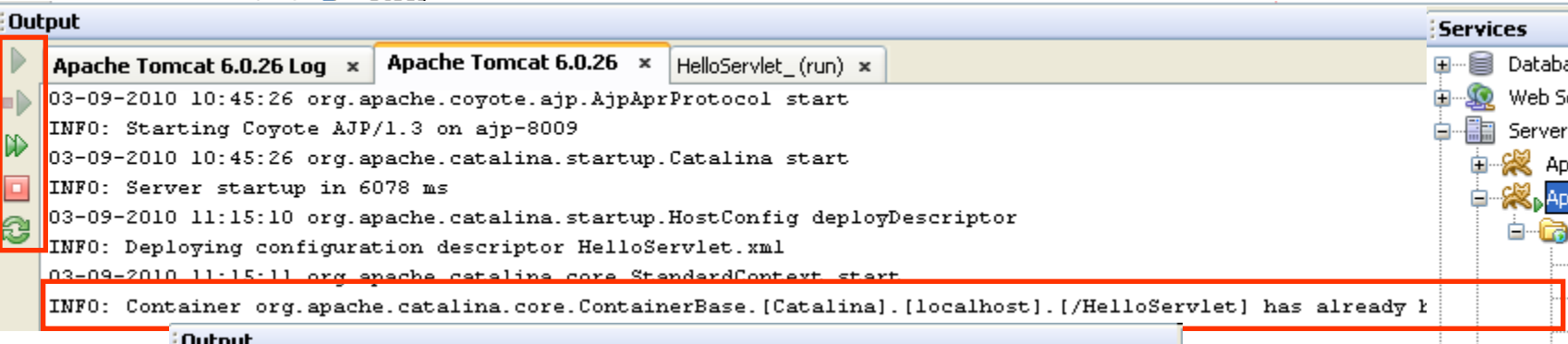
Web Applications

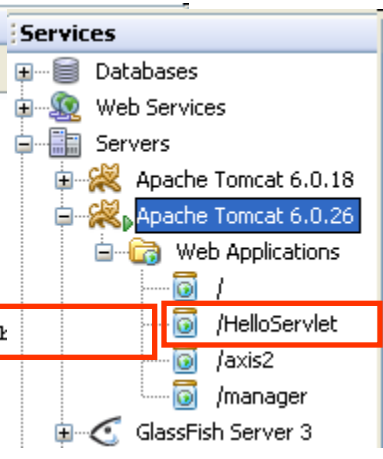
Web Application Development Process

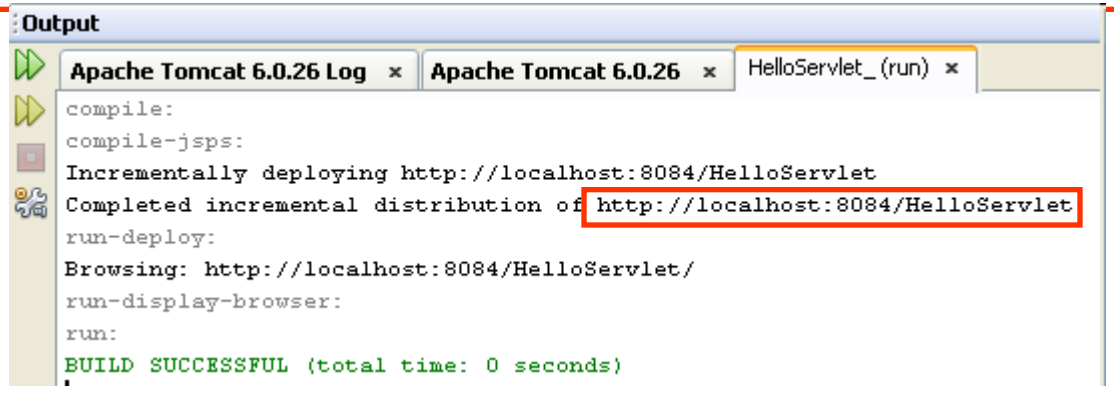
- **Step 4, 5 & 6:** Building, Deploying & Executing









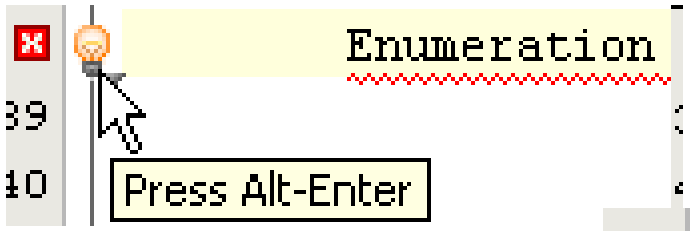


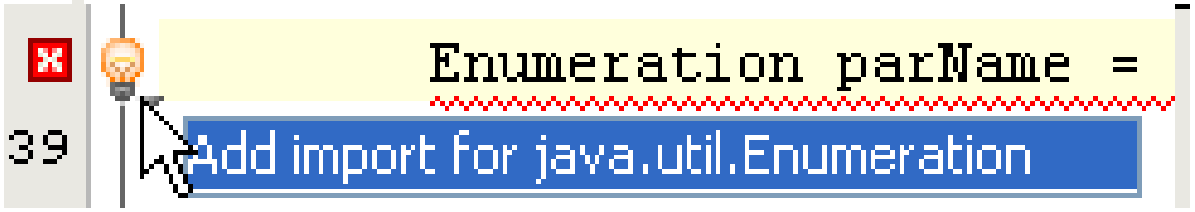
Web Applications

Web Application Development Process

- **Step 5 & 6:** Deploying & Executing

Web Applications
Additional





- **Caches of server**
  - **WinXP**: C:\Documents and Settings**LoggedInUser**\.netbeans\6.9\apache-tomcat-6.0.26_base\work\Catalina\localhost\
  - **Vista or Win7**: C:\Users**LoggedUser**\.netbeans\6.9\apache-tomcat-6.0.26_base\work\Catalina\localhost\
  - Above location should be gone and cleared when the application cannot be undeployed or the web servers occur the errors

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** - Initialises 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
getInitParameter	- **public String getInitParameter(String name)** - Return a String containing the value of name initialisation
getServletContext	- **public ServletContext getServletContext()** - Returns the ServletContext in which this servlet instance is running
getServletConfig	- **public ServletConfig getServletConfig()** - returns the servlet configuration objects of this servlet instance
getServletInfo	- **public String getServletInfo()** - returns the useful servlet information on the name of the creator of the servlet, version information and copyright information

Servlet Life Cycle defined in Generic

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”);

The Servlet Model

ServletRequest interface

Methods	Descriptions
getAttribute	- **public Object getAttribute(String name)** - Retrieves the **value of an attribute specified by the name**, that was set using the setAttribute() method. - Returns **null when no attribute with the specified name exists** - String strUser = (String)request.getAttribute(”Username”)
getContentLength	- **public int getContentLength()** - Returns the **length of content in bytes & return -1(length isn’t know)**
getInputStream	- **public ServletInputStream getInputStream() throws IOException** - Returns the **binary data of the body of request requested** by the client and **stores** it **in** a **ServletInputStream object** - ServletInputStream inStr = request.getInputStream();
getServerName	- **public String getServerName()** - returns the **host name of the server** to which the client request was sent - String serverName = request.getServerName();
setCharacterEncoding	-**public void setCharacterEncoding (String env)** -**Overrides** the name of the character encoding used in the body of this request. This method must be called **prior to reading** request parameters or reading input using getReader(). Otherwise, it has no effect.

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

HttpServlet class

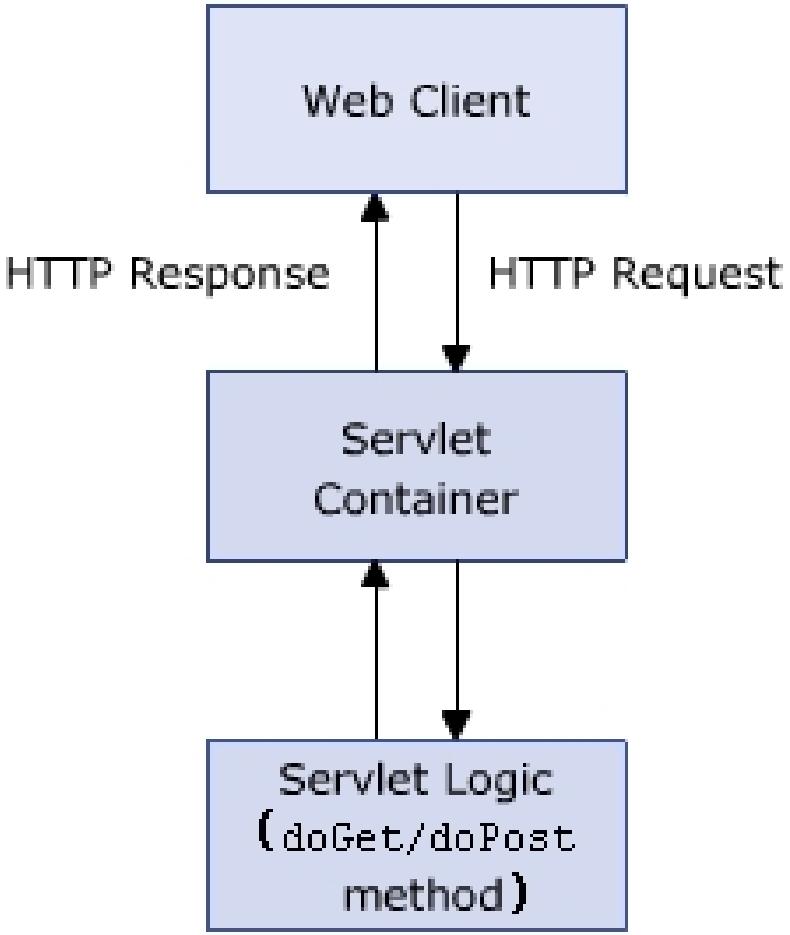
- The protocol **defines** a set of **text-based request messages** called HTTP 'methods' **implemented in *HttpServlet* class**
- Provides **an abstract class** to **create** an **HTTP Servlet**
- **Extends** the **GenericServlet class**
- A subclass of HttpServlet class **must override at least** one 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



The Servlet Model

HttpServletRequest interface

Methods	Descriptions
getCookies	- **public Cookie[] getCookies()** - Returns an **array containing** the entire **Cookie objects** - Returns **null** if no cookies were found - Ex: Cookie[] cookie = request.getCookies();
getMethod	- **public String getMethod()** - Returns a **name** of the **HTTP method** used to make the request. - Ex: String method = request.getMethod();
getPathInfo	- **public String getPathInfo()** - Returns the **path information associated** with a **URL**. - Ex: String strPath = request.getPathInfo();
getAuthType	- **public String getAuthType()** - Returns the **basic authentication schema** used to protect the servlet from unauthorized users

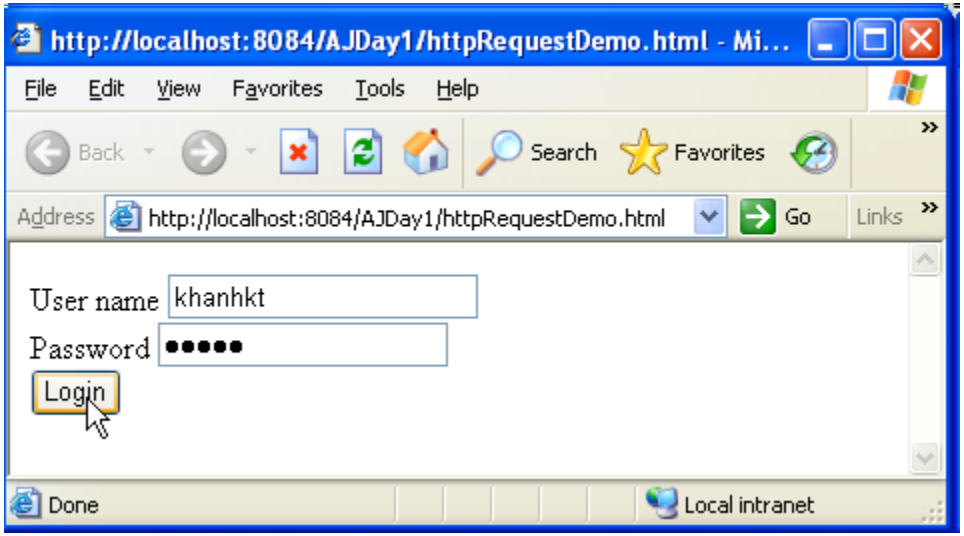
The Servlet Model

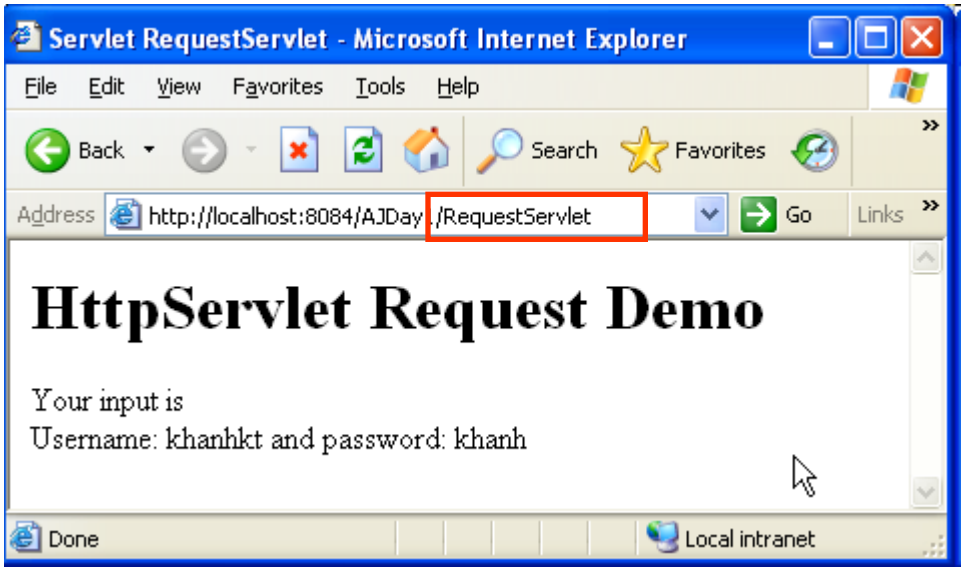
HttpServletRequest interface

Methods	Descriptions
getHeader	- **public String getHeader(String name)** - Returns the **value** of the **specified request header as a String**. - Returns **null** if the request did not include a header name - Ex: String strHost = request.getHeader(”host”);
getHeaders	- **public Enumeration getHeaders(String name)** - returns **all values of the specified request header** as an **Enumeration of String** objects - Request Header Allow the client to pass additional information about request, client itself to the server **Some request headers** •**Accept**: specifies **types of headers acceptable** by client •**Accept – Charset**: the **character sets acceptable** by the response •**Accept – Encoding**: **restriction** of **content-coding** which is accepted by response •**Accept – Language**: **restriction** of **natural languages** which is used by response •**Authorization**: **authentication** of a user agent with a server - Ex String headers = request.getHeaders(“Accept”);
getHeaderNames	- **public Enumeration getHeaderName()** - returns an **enumeration** of **all** the **header name** - returns **empty** enumeration if the request has no headers - Ex: String headers = request.getHeaders();

The Servlet Model

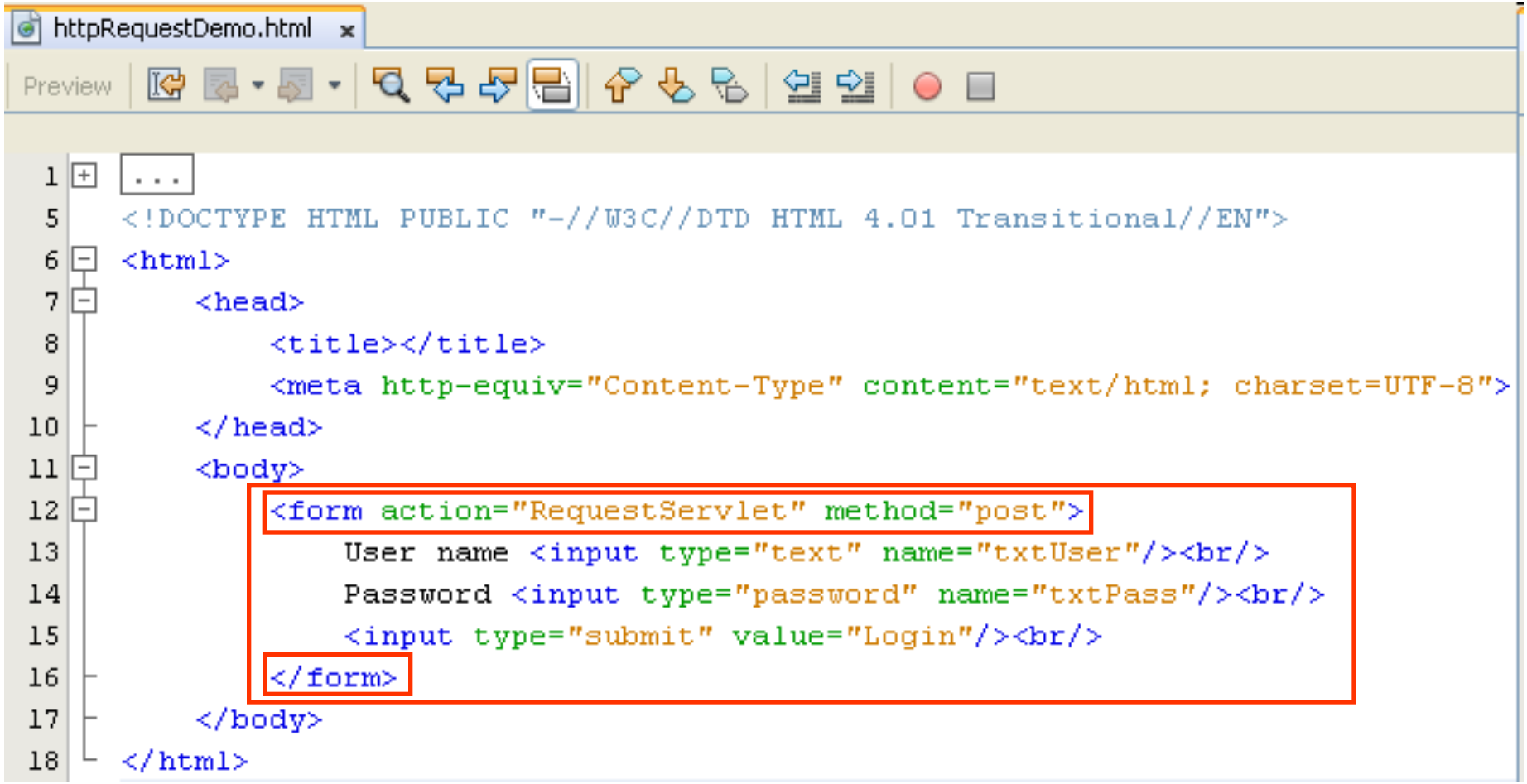
HttpServletRequest interface – Examples





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>
```

The Servlet Model

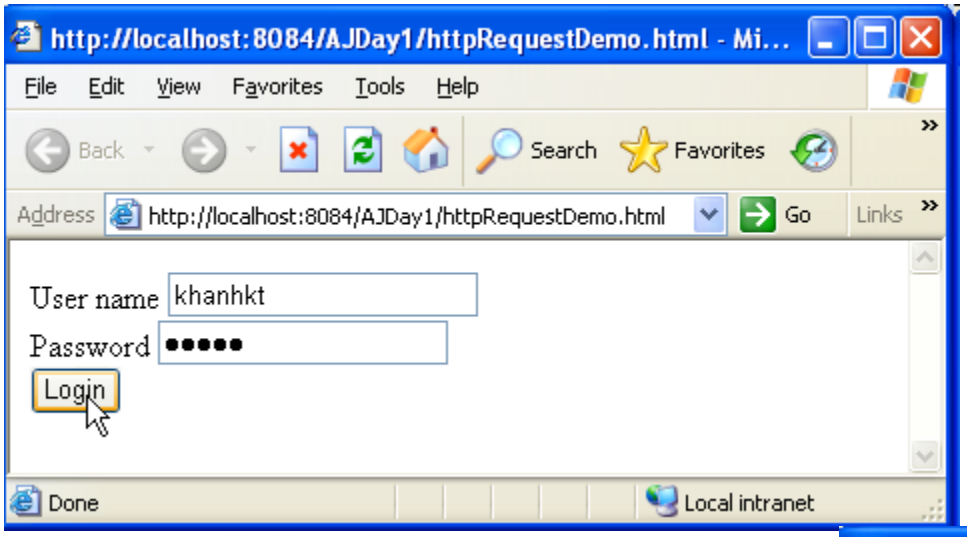
HttpServletRequest interface – Examples

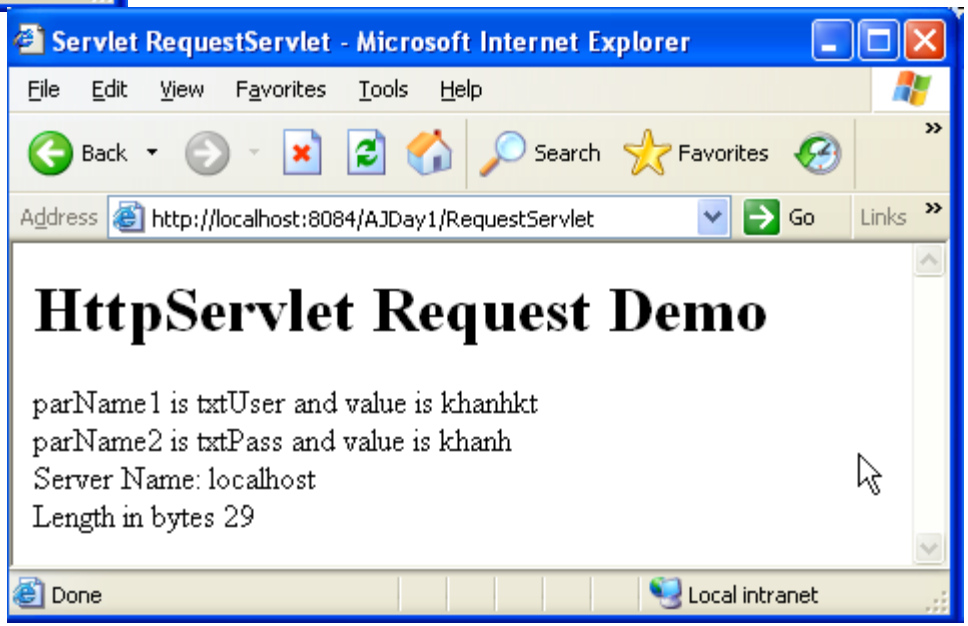
RequestServlet.java

```
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              out.println("<html>");
34              out.println("<head>");
35              out.println("<title>Servlet RequestServlet</title>");
36              out.println("</head>");
37              out.println("<body>");
38              out.println("<h1>HttpServlet Request Demo</h1>");
39              String username = request.getParameter("txtUser");
40              String password = request.getParameter("txtPass");
41
42              out.println("Your input is <br/>");
43              out.println("Username: " + username + " and password: " + password);
44
45              out.println("</body>");
46              out.println("</html>");
47          } finally {
48              out.close();
49          }
50      }
```

The Servlet Model

HttpServletRequest interface – Examples





The Servlet Model

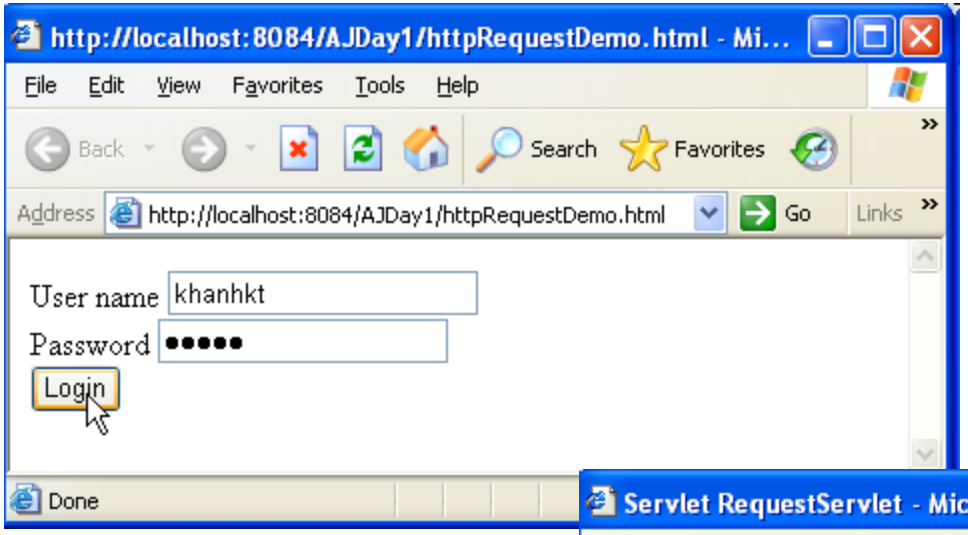
HttpServletRequest interface – Examples

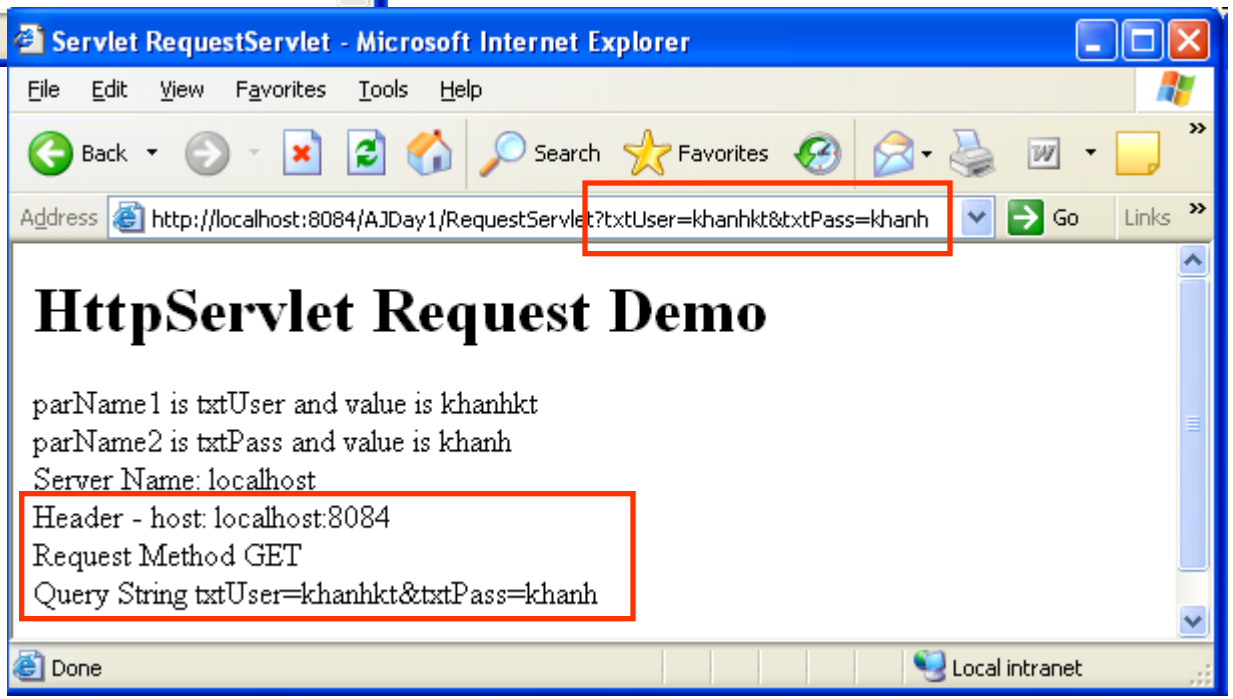
RequestServlet.java

```
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              out.println("<html>");
34              out.println("<head>");
35              out.println("<title>Servlet RequestServlet</title>");
36              out.println("</head>");
37              out.println("<body>");
38              out.println("<h1>HttpServlet Request Demo</h1>");
39              Enumeration parNames = request.getParameterNames();
40              int count = 0;
41              while (parNames.hasMoreElements()) {
42                  ++count;
43                  String parName = (String) parNames.nextElement();
44                  out.print("parName" + count + " is " + parName);
45
46                  String parVal = request.getParameter(parName);
47                  out.println(" and value is " + parVal + "<br/>");
48              }
49              String strServer = request.getServerName();
50              out.println("Server Name: " + strServer + "<br/>");
51              int length = request.getContentLength();
52              out.println("Length in bytes " + length + "<br/");
53              out.println("</body>");
54              out.println("</html>");
55          } finally {
56              out.close();
57          }
58      }
```

The Servlet Model

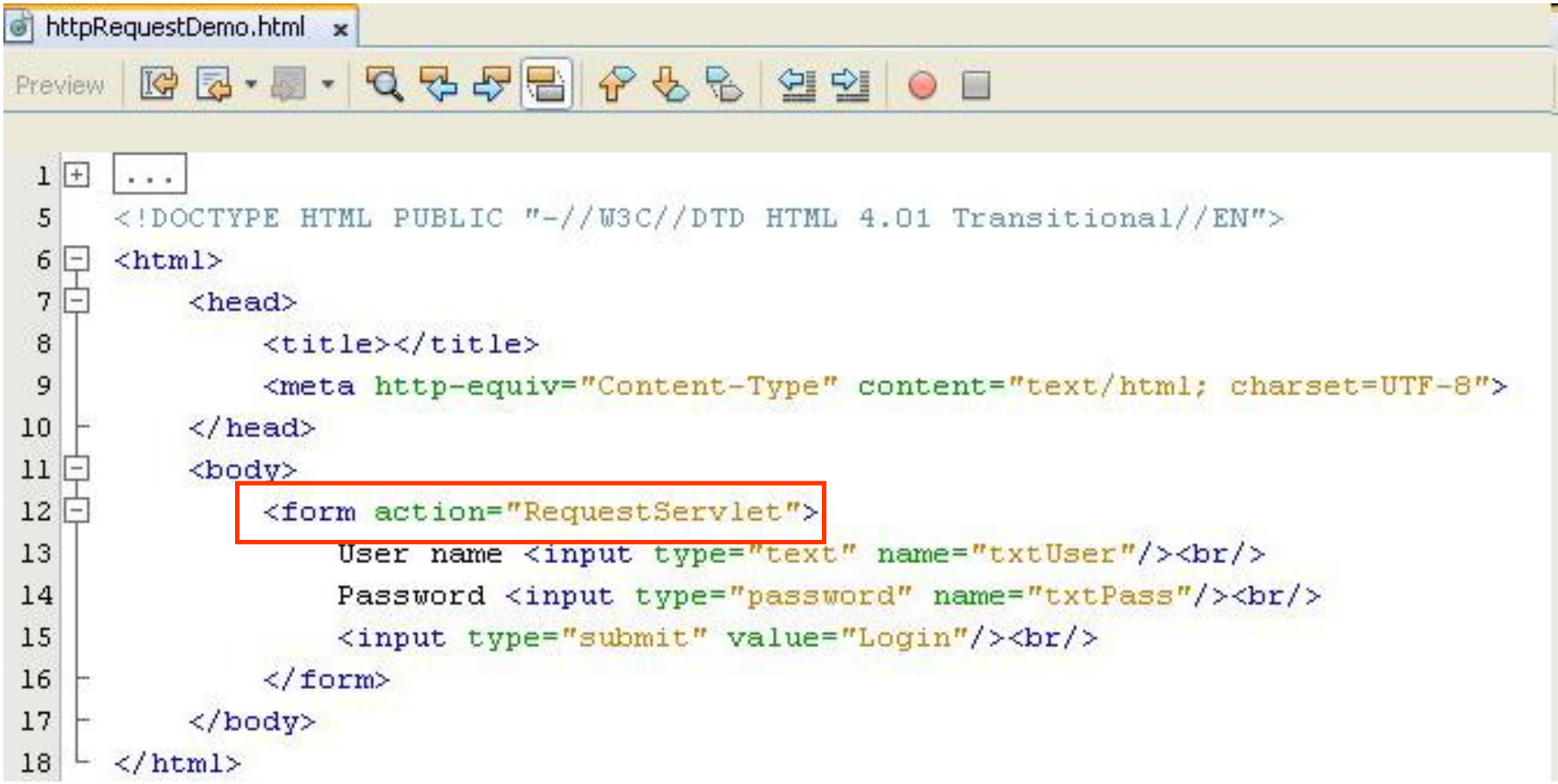
HttpServletRequest interface – Examples





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>
```

The Servlet Model

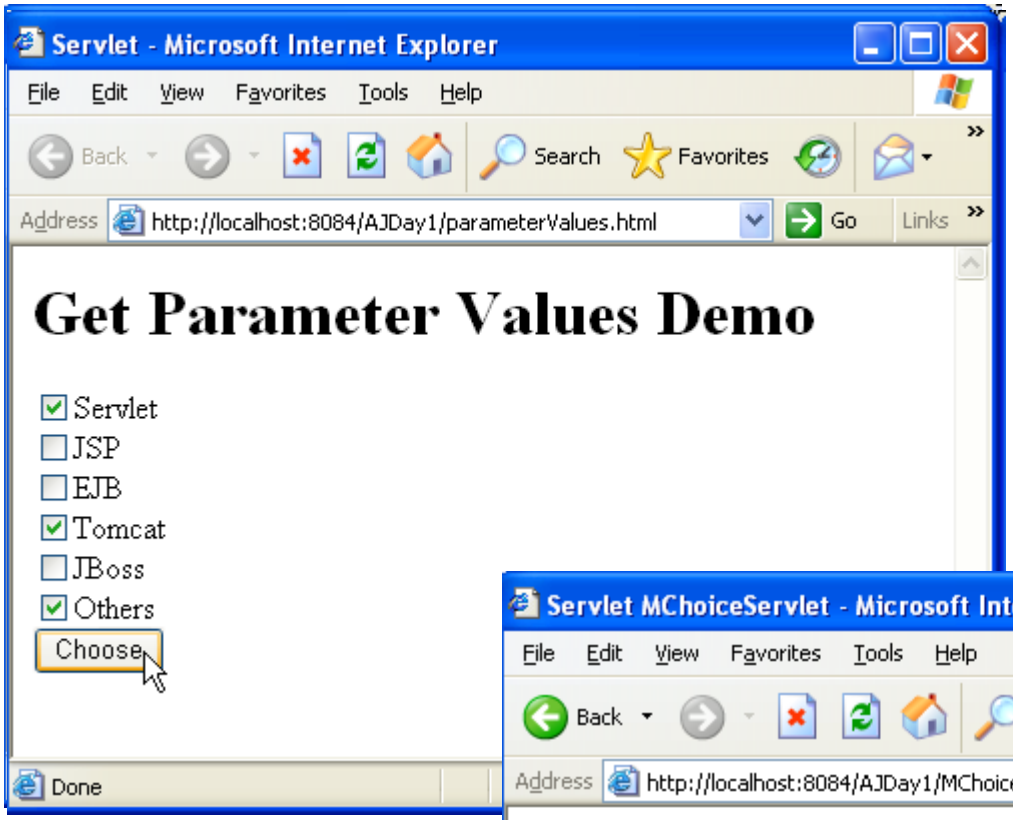
HttpServletRequest interface – Examples

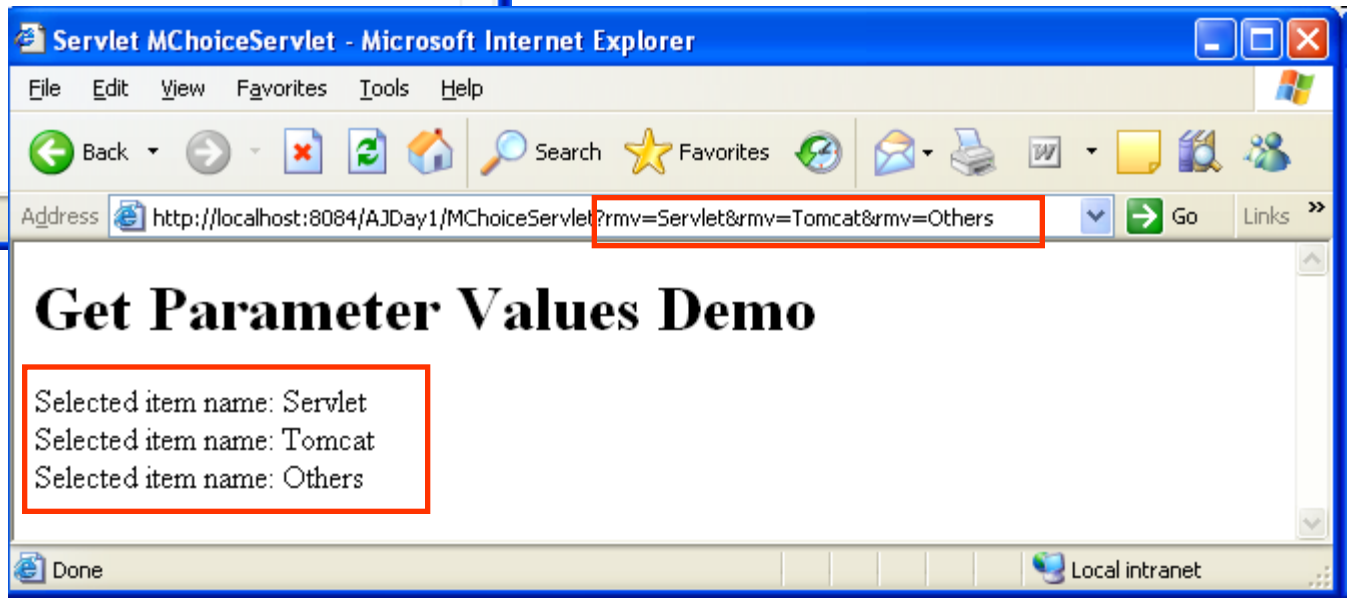
RequestServlet.java

```
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>HttpServlet Request 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>");
```

The Servlet Model

HttpServletRequest interface – Examples





The Servlet Model

HttpServletRequest interface – Examples

parameterValues.html

```
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>
```

The Servlet Model

HttpServletRequest interface – Examples

MChoiceServlet.java

```
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      }
```

The Servlet Model

HttpServletRequest interface – Examples

```html
<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>
```

multiparam.html | Controller.java

```java
28      protected void processRequest(HttpServletRequest request, HttpServle
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      }
```

The Servlet Model

HttpServletRequest interface – Examples

Address http://localhost:8084/Day1Servlet1005Y2/multiparam.html

Num1 1

Num2 2

Num3 3

Perform

Address http://localhost:8084/Day1Servlet1005Y2/Controller?txtNum=1&txtNum=2&txtNum=3

Multiparam Demo

Num is 1 - 1 - 1

The Servlet Model

HttpServletRequest interface – Examples

multiparam.html | Controller.java

```
28      protected void processRequest(HttpServletRequest request, Htt
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      }
```

Address http://localhost:8084/Day1Servlet1005Y2/Controller?txtNum=1&txtNum=2&txtNum=3

Multiparam Demo

Num is 1, 2, 3,

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
addCookies	- **public void addCookie(Cookie cookie)** - **Adds** specified **cookie to** the **response** sent to the client - response.addCookie(new Cookie("Aptech", "Servlet");
sendError	- **public void sendError(int sc) throws IOException** - **Send** an **error response to** the **client using** the **specified status** code and clearing the buffer - response.sendError(HttpServletResponse.SC_FORBIDDEN, "Goodbye");
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

Methods	Descriptions
addHeader	- **public void addHeader(String name, String value)** - **Add name and value to** the **response** header. - Ex: response.addHeader(”Refresh”, 15); - Response header + is attached to the files being sent back to the client + contains the date, size and type of file that server sends back to the client and also data about the server itself + can be used to specify cookies to supply the modification date + used to instruct the browser to reload the page + it specifies how big the file is, to dertermine how long the HTTP connection needs to be maintained
containHeader	- public boolean containsHeader(String header)–return true if the response header has any values - Verify if the response header **contains any values.** - Returns true if the response header has any values. Otherwise, returns false - Ex: response.containsHeader(“Cache”);
addDateHeader	- **public void addDateHeader(String name, long date)** - Adds response header with the **given name and date value** - Ex: response.addDateHeader(“Cache”, 20-02-2002)
addIntHeader	- **public void addIntHeader(String name, int value)** - Adds response header with the given **name and integer value** - Ex: response.addIntHeader(“Chache”, 3)
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”);

The Servlet Model

HttpServletResponse interface - Example

- Using sendRedirect

RedirectServlet.java

```
14  /**
15   *
16   * @author Trong Khanh
17   */
18  public class RedirectServlet 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              response.sendRedirect(response.encodeRedirectURL("ResServlet"));
33          } finally {
34              out.close();
35          }
36      }
```

The Servlet Model

HttpServletResponse interface - Example

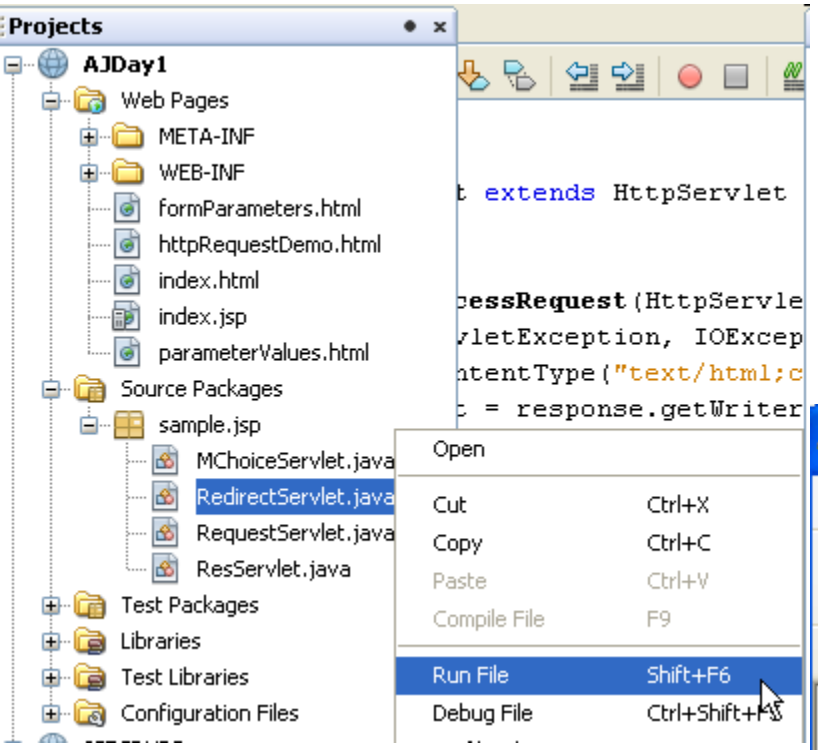
- ResServlet

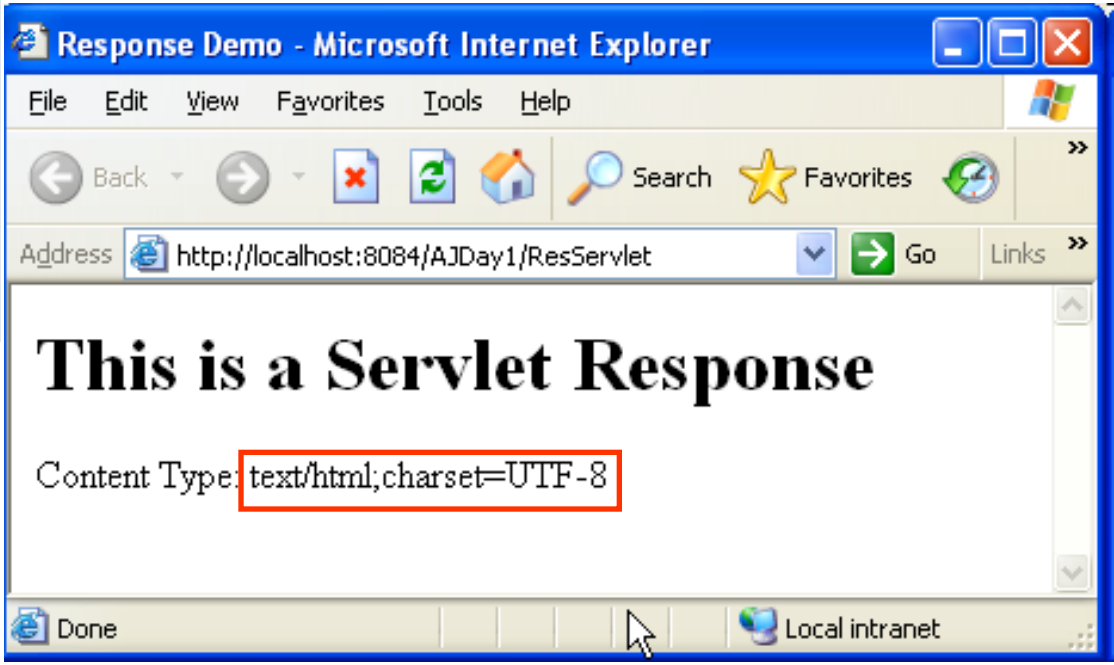
ResServlet.java

```
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

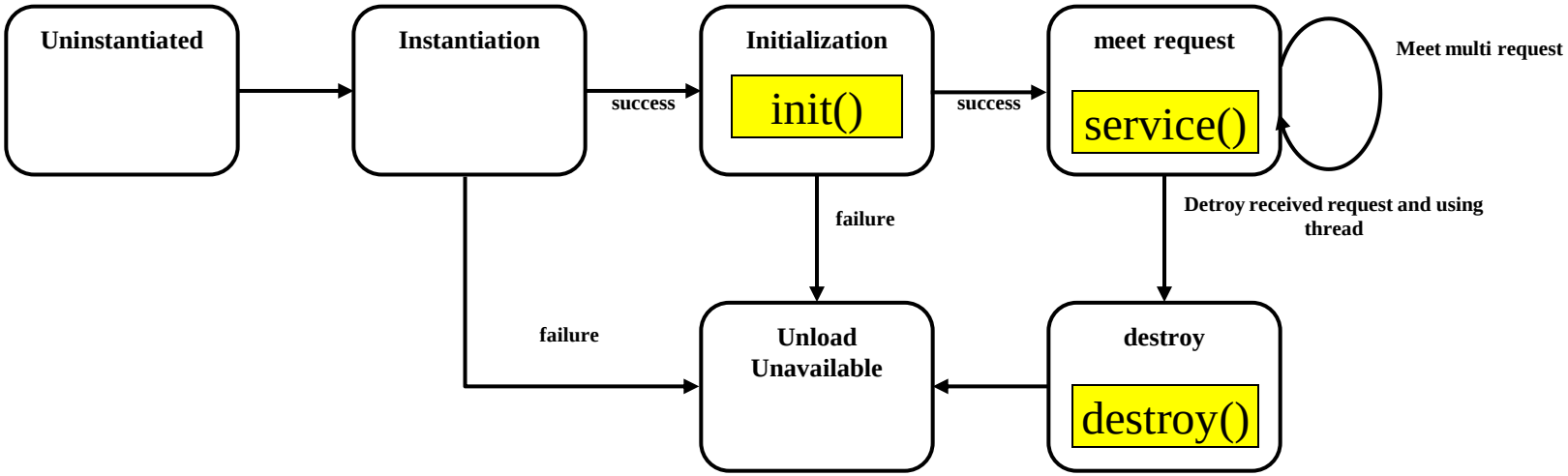
HttpServletResponse interface - Example





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

```
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

LifeCycleServlet.java

```
52      // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the
53      /**...*/
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
61              throws ServletException, IOException {
62          processRequest(request, response);
63          System.out.println("doGet is invoked");
64      }
65
66      /**...*/
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
74              throws ServletException, IOException {
75          processRequest(request, response);
76          System.out.println("doPost is invoked");
77      }
```

Servlet - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites

Address http://localhost:8084/AJDay1/LifeCycleServlet

Servlet Life Cycle

a = 25

Done

Output

Apache Tomcat 6.0.26 Log | Apache Tomcat 6.0.26 | AJDay1 (run)

```
INFO: Container org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/AJDay1] has
init
a = 5
doGet is invoked
doGet is invoked
```

The Servlet Model

The Servlet Life Cycle – Example

LifeCycleServlet.java

```
16    * @author Trong Khanh
17    */
18   public class LifeCycleServlet extends HttpServlet {
19       private int a = 0;
         public void init() throws ServletException {...}
26       /**...*/
33       protected void processRequest(HttpServletRequest request, HttpServletResponse response)
34               throws ServletException, IOException {...}
51
         protected void service(HttpServletRequest request, HttpServletResponse response)
53               throws ServletException, IOException {
54           System.out.println ("service");
55           response.setContentType("text/html");
56           PrintWriter out=response.getWriter();
57           out.println("This is service");
58       }
```

http://localhost:8084/AJDay1/LifeCycleServlet - Microso...

File Edit View Favorites Tools Help

Back Search

Address http://localhost:8084/AJDay1/LifeCycleServlet

This is service

Done

Output

Apache Tomcat 6.0.26 Log | Apache Tomcat 6.0.26 | AJDay1 (run)

```
21-06-2011 17:19:32 org.apache.catalina.core.StandardContext start
INFO: Container org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/AJDay1]
init
a = 5
service
```

The Servlet Model

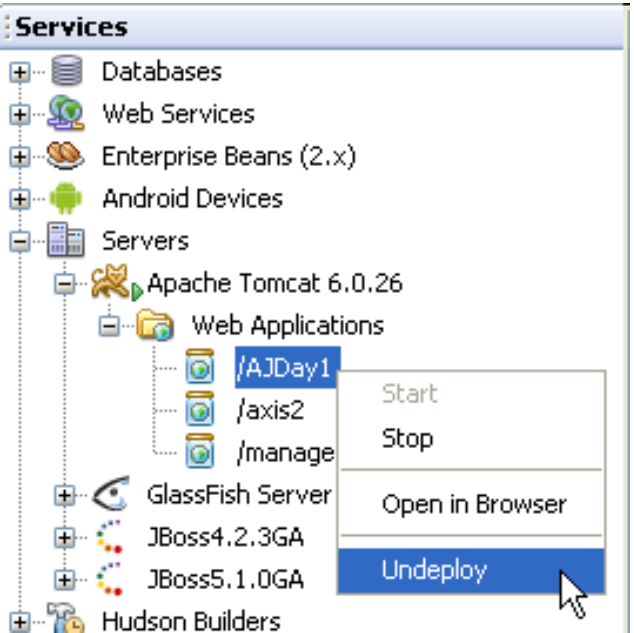
The Servlet Life Cycle – Example

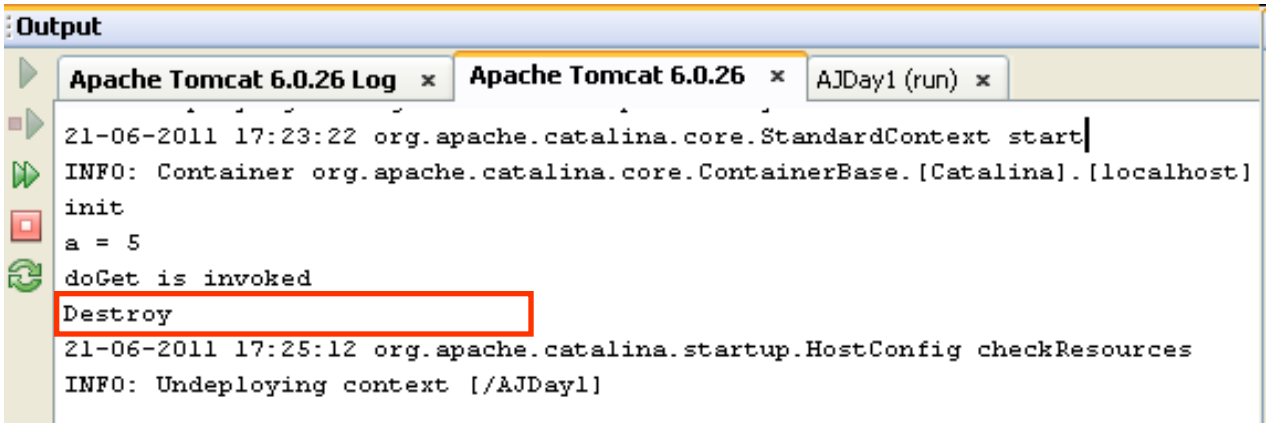
- **Addition the destroy method (**comment service method**)**

LifeCycleServlet.java

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

- **Execute project again, then undeploy the current project on Tomcat Server**

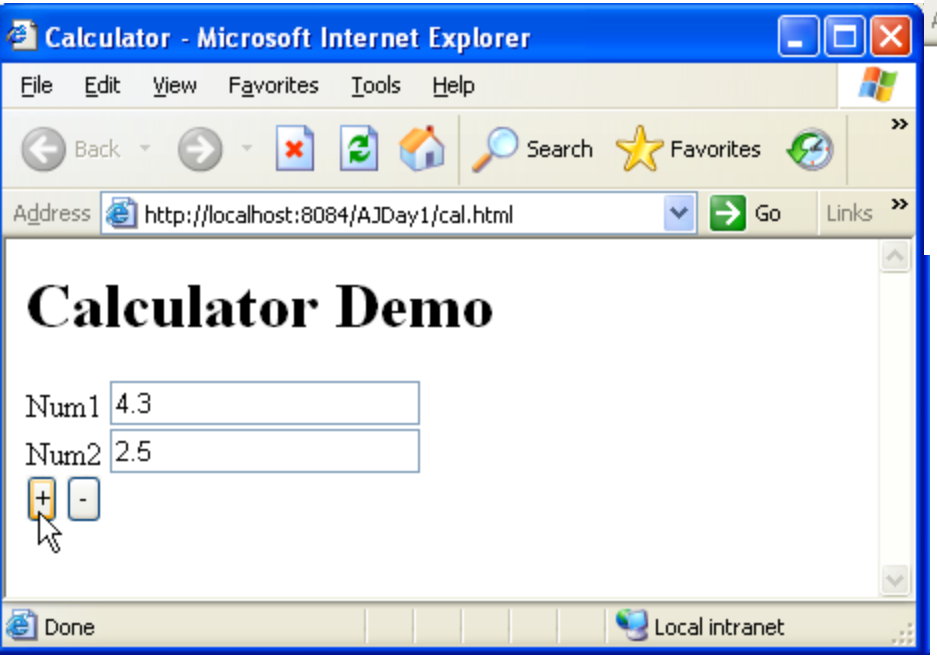


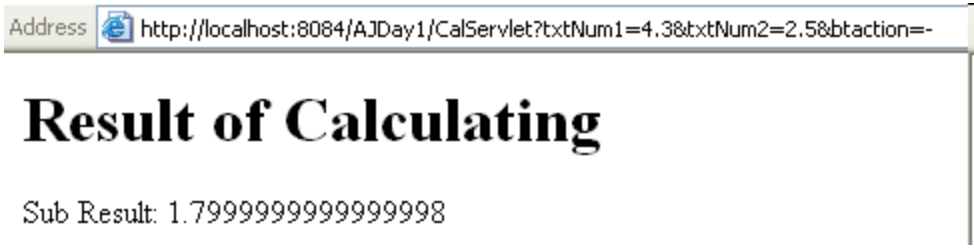


The Servlet Model

Example

- Building the web application can do some following function
  - The application allows the user calculating the add and subtract operation of 2 numbers that are input from the user interface
  - The result of calculating will be presented after the user press the corresponding button





The Servlet Model

Example

- **Form parameter using html should be implemented as following**

cal.html

```
1  ...
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
6  <html>
7    <head>
8      <title>Calculator</title>
9      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
10   </head>
11   <body>
12       <h1>Calculator Demo</h1>
13       <form action="CalServlet">
14           Num1 <input type="text" name="txtNum1" value="" /><br/>
15           Num2 <input type="text" name="txtNum2" value="" /><br/>
16           <input type="submit" value="+" name="btaction" />
17           <input type="submit" value="-" name="btaction" />
18       </form>
19   </body>
20 </html>
```

The Servlet Model

Example

- **CalServlet should be implemented as following**

CalServlet.java

```
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>Calculating</title>");
35              out.println("</head>");
36              out.println("<body>");
37              out.println("<h1>Result of Calculating</h1>");
38
39              String action = request.getParameter("btaction");
40              if (action.equals("+")) {
41                  String num1 = request.getParameter("txtNum1");
42                  String num2 = request.getParameter("txtNum2");
43                  double n1 = Double.parseDouble(num1);
44                  double n2 = Double.parseDouble(num2);
45                  out.println("Add Result: " + (n1 + n2));
46              } else if (action.equals("-")) {
47                  String num1 = request.getParameter("txtNum1");
48                  String num2 = request.getParameter("txtNum2");
49                  double n1 = Double.parseDouble(num1);
50                  double n2 = Double.parseDouble(num2);
51                  out.println("Sub Result: " + (n1 - n2));
52              }
53              out.println("</body>");
54              out.println("</html>");
55          } catch (NumberFormatException e) {
            e.printStackTrace();
57          } finally {
```

Summary

- **HTML Introduction**
- **The Servlet Model**

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

- **Web Application**
  - Web application Structure
  - Web Deployment Descriptors
- **The Web Container Model**
  - Attribute, Scope (Request, Session, Application)
  - Request Dispatching
  - Filters