UNIT-6

Prepared By:
Deepak Kumar Sharma
Asst. Professor
SoCS UPES Dehradun

UNIT-VI

| |
|---|
| Servlets: Introduction, Benefits, Architecture, GET, POST methods, Servlet container, Servlet's Life Cycle |
| Generic servlet, HTTP servlet (HttpServlet Class, HttpServletRequest, HttpServletResponse interface) |
| ServletConfig, ServletContext, Requests & Responses, GenericServlet, Thread-Safe Servlets |
| Deployment Descriptor, Session Tracking/Management (Cookies, Hidden Form Fields, URL Rewriting, HttpSession & Session Objects) |
| Servlet Filter, Servlet Listeners |
| Problem with servlets, JSP: Introduction, How JSP work, Architecture, Lifecycle, Directives (Page, Include, TagLib) |
| |

Introduction to Servlet

- We define a Java Servlet or Jakarta Servlet as the technology to design and deploy dynamic web pages using the Java Programming Language.

- It implements a typical servlet in the client-server architecture, and the Servlet lives on the server-side.

- Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology.
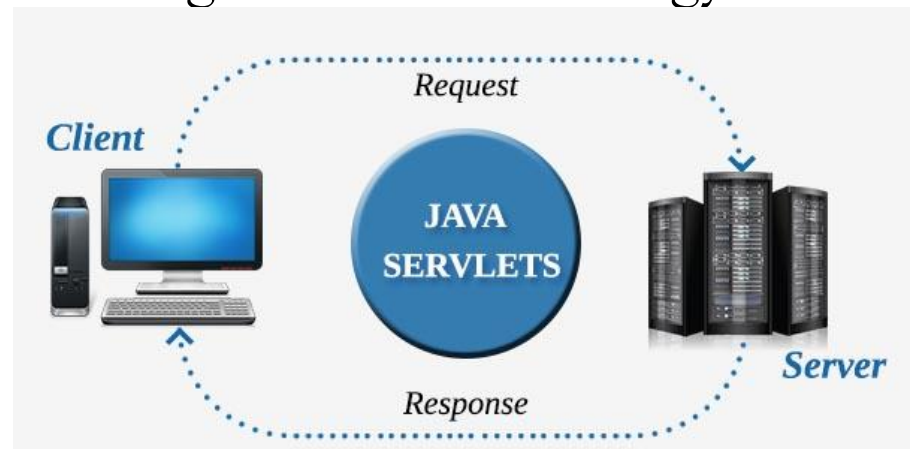
What is Servlet?

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.
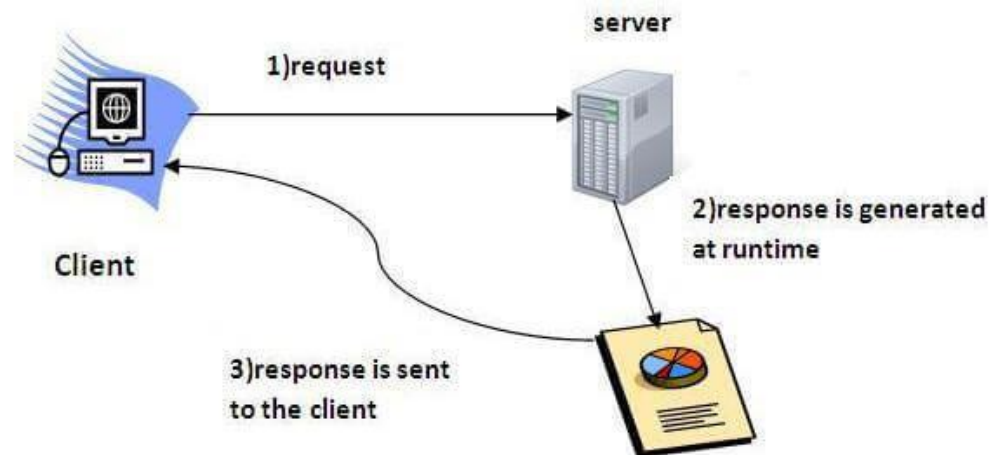
What is a web application?

- A web application is an application accessible from the web.

- A web application is composed of web components like Servlet, JSP, Filter, etc. and other elements such as HTML, CSS, and JavaScript.

- The web components typically execute in Web Server and respond to the HTTP request.

Java Servlet Features

- Portable
  - The Servlet program designed in one Operating System's Platform can be run in a different Operating System Platform with ease.

- Efficient
  - It performs in every environmental condition regardless of the operating system platform.

- Scalable
  - Servlets use completely lightweight threads for the processes and can simultaneously handle multiple client requests by generating various threads.

- Robust
  - The servlets extend the features of Java:
    - Java Security Manager
    - Java Garbage Collector
    - Exception Handling.

Advantages of Servlets

- Servlets are server independent

- Servlets are protocol-independent, i.e. it supports FTP, SMTP, etc. Mainly it provides extended support to HTTP protocol functionality.

- Servlets are persistent because it remains in memory until explicitly destroyed this helps in several request processing and one database connection can handle several database requests.

- Servlets are portable; since the servlets are written in java, they are portable and supports any web server.

- Faster in execution, servlets are compiled into byte code therefore execute more quickly compared to other scripting languages. Byte code conversion gives better performance and helps in type checking and error.

Deepak Sharma, Asst. Professor UPES Dehradun

Architecture

- Components of Servlet Architecture
  1. Client
  2. Web Server
  3. Web Container

Architecture

1. Client

- The web browser acts as a Client.

- Client or user connected with a web browser.

- The client is responsible for sending requests or HttpRequest to the web server and processing the Web server's responses.

Architecture

2. Web Server

- Web server controls how web user access hosted files, and it's responsible for processing user request and responses.

- Server understands URLs and HTTP protocol.

- Whenever a browser needs to host a file on the webserver, process client request using an HTTP request; if it finds the requested file sends it back to the browser through HTTP Response.

- There are two types' web servers Static and Dynamic webservers.
  - in a static web server, it sends the file as it is, but in a dynamic web, the server-hosted file is updated before it is sent to the browser.

Architecture

3. Web Container/Servlet Container

- Web container handles the requests of servlets, JSP and other files at the server-side.

- A web container is responsible for managing the lifecycle of servlets, and it also performs the URL mapping task.

- The important tasks performed by servlets are loading and unloading servlets, creating and managing requests and response objects, and performing servlet management's overall task.

Servlet Request Flow

- The client sends a request.

- Web Server accepts the request and forwards it to the web container.

- Web container searches web.xml file for request URL pattern and gets the address of the servlet.

- If the servlet is not yet instantiated, it will be instantiated and initialized by calling the init() method.

- The container calls public service() by passing ServletRequest and ServletResponse objects.

- Public service() method typecast ServletRequest and ServletResponse object to HttpServletRequest and HttpServletResponse objects respectively.

- Public service() method calls protected service().

- Protected service() method checks the client request & corresponding do____() method is called.

- The request is handled by sending the result generated by do____() to the client.

GET and POST Methods

- HTTP
  - Request/response protocol which is based on client/server based architecture.
  - Web browser, search engines, etc. behave as HTTP clients and the Web server like Servlet behaves as a server.

- Two common methods for the request-response between a server and client are:
  - **GET**- It requests the data from a specified resource
  - **POST**- It submits the processed data to a specified resource

Get Request

- The query string (name/value pairs) is sent inside the URL of a GET request:

  GET/RegisterDao.jsp?name1=value1&name2=value2

  - Data is sent in request header.

- Some other features of GET requests are:

  - It remains in the browser history

  - It can be bookmarked

  - It can be cached

  - It has length restrictions

  - It should never be used when dealing with sensitive data

  - It should only be used for retrieving the data

Post Request

- The query string (name/value pairs) is sent in HTTP message body for a POST request.

- Requests cannot be bookmarked

- Requests have no restrictions on length of data

- Requests are never cached

- Requests do not retain in the browser history

Get vs. Post

GET	POST
values are visible in the URL.	values are not visible in the URL.
limitation on the length of the values, generally 255 characters.	no limitation on the length of the values since they are submitted via the body of HTTP.
Better performance because of the simple nature of appending the values in the URL.	It has lower performance as compared to GET method because of time spent in including POST values in the HTTP body.
This method supports only string data types.	This method supports different data types, such as string, numeric, binary, etc.
GET results can be bookmarked.	POST results cannot be bookmarked.
GET request is often cacheable.	The POST request is hardly cacheable.
GET Parameters remain in web browser history.	Parameters are not saved in web browser history.

Java Servlet Life-Cycle

- The web container maintains the life cycle of a servlet instance.

- Three states:
  - New- init()
  - Ready- service()
  - End- destroy()

1.Load servlet class

2.Create servlet instance

3.Call the init(-) method

4.Call the service(-,-) method

READY

5.Call the destroy() method

Servlet Container

Servlet

Load and Instantiate servlet

init()

service()
doGet(req,resp)
doPost(req,resp)

destroy()

Thread 1

Thread 2

HTTP Response

HTTP Request

HTTP Response

HTTP Request

Web Server

Deepak Sharma, Asst. Professor UPES Dehradun
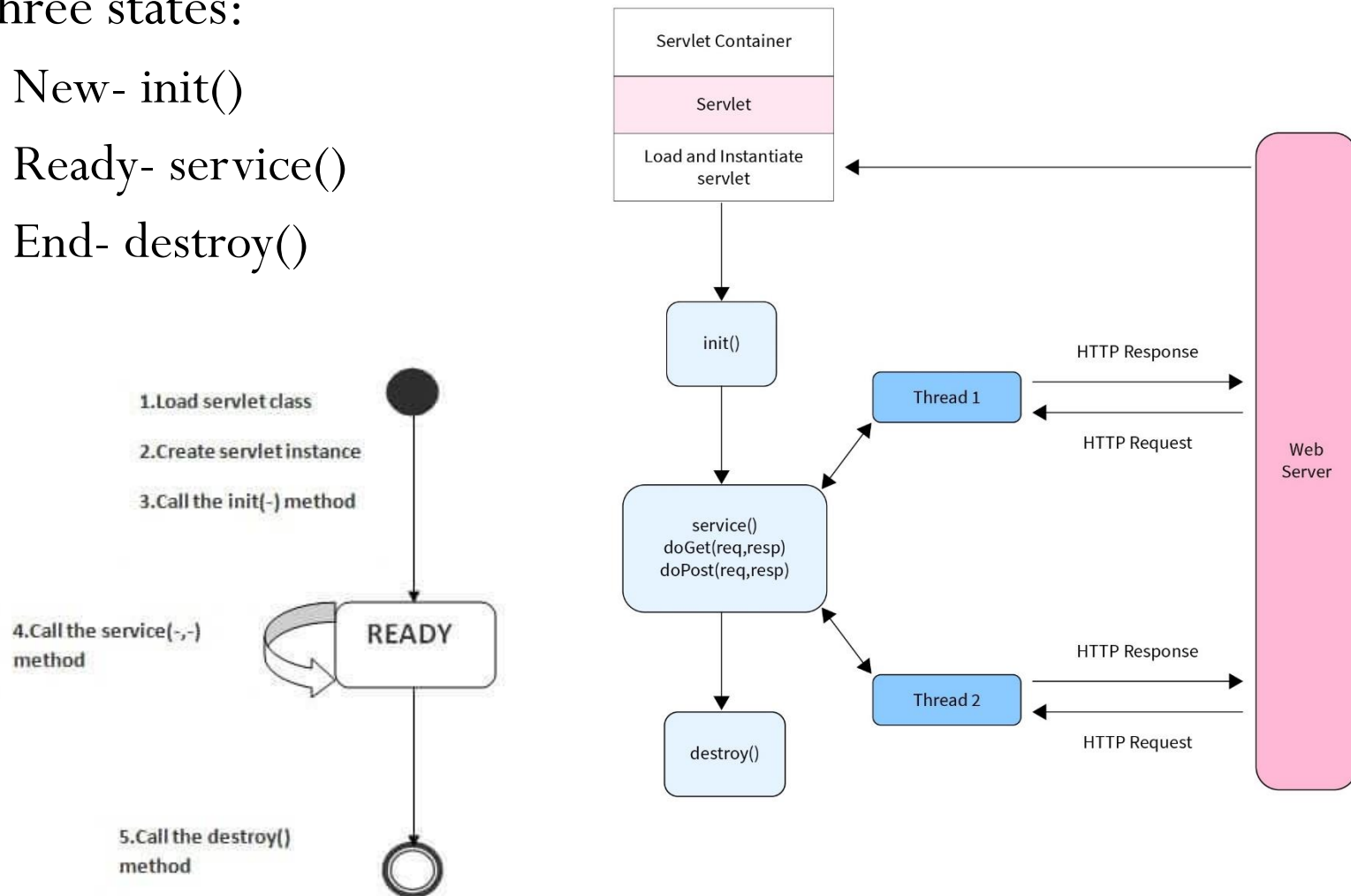
Java Servlet Lifecycle – new state or Init()

1) Servlet class is loaded
- The classloader is responsible to load the servlet class.
- The servlet class is loaded when the first request for the servlet is received by the web container.

2) Servlet instance is created
- The web container creates the instance of a servlet after loading the servlet class.
- The servlet instance is created only once in the servlet life cycle.

3) init method is invoked
- The web container calls the init method only once after creating the servlet instance.
- The init method is used to initialize the servlet.
- It is the life cycle method of the javax.servlet.Servlet interface.

Syntax of the init method:

public void init(ServletConfig config) **throws** ServletException

Java Servlet Lifecycle – ready or service()

- The web container calls the service method each time when request for the servlet is received.

- If servlet is not initialized, it follows the first three steps of new state. If servlet is initialized, it calls the service method.

Syntax:
public void service(ServletRequest request, ServletResponse response) **throws** ServletException, IOException

- The client may request various services like:
  - GET, PUT, UPDATE & DELETE

- The service() method takes responsibility to check the type of request received from the client and respond accordingly by generating a new thread or a set of threads as per the requirement and implementing the operation through the following methods.
    - doGet() for GET
    - doPut() for PUT
    - doUpdate() for UPDATE
    - doDelete() for DELETE

Java Servlet Lifecycle – End or destroy()

- The web container calls the destroy method before removing the servlet instance from the service.

- It gives the servlet an opportunity to clean up any resource for example memory, thread etc.

- The syntax of the destroy method of the Servlet interface is given below:

  public void destroy()

Generic Servlet Class

- GenericServlet implements the Servlet interface and provides an implementation for all its methods except the **service() method.**

- GenericServlet class defines a protocol-independent(HTTP-less) servlet.

- However, while building a website or an online application, we may want to have HTTP protocol, in that case, we must extend HttpServlet instead of GenericServlet.

- Developing Servlet by extending GenericServlet is very easy because we have to provide implementation only for the **service()** method.

- GenericServlet class is in javax.servlet package (javax.servlet.GenericServlet).

Methods of GenericServlet class

- **public void init(ServletConfig config)** is used to initialize the servlet.

- **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.

- **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.

- **public ServletConfig getServletConfig()** returns the object of ServletConfig.

- **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.

- **public void init()** it is a convenient method for the servlet programmers, now there is no need to call super.init(config)

Methods of GenericServlet class

- **public ServletContext getServletContext()** returns the object of ServletContext.

- **public String getInitParameter(String name)** returns the parameter value for the given parameter name.

- **public Enumeration getInitParameterNames()** returns all the parameters defined in the web.xml file.

- **public String getServletName()** returns the name of the servlet object.

- **public void log(String msg)** writes the given message in the servlet log file.

- **public void log(String msg,Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

Service() method

- This method is automatically called by the server whenever a request for a GenericServlet arrives.

  public void service (ServletRequest req, ServletResponse resp) throws ServletException,IOException

- The ServletRequest object allows to read data provided by the client request and
- the ServletResponse object is used to send the response to the client

Servlet Example by inheriting the GenericServlet class

```java
import java.io.*;
import javax.servlet.*;

public class First extends GenericServlet{
public void service(ServletRequest req,ServletResponse res)
throws IOException,ServletException{

res.setContentType("text/html");

PrintWriter out=res.getWriter();
out.print("<html><body>");
out.print("<b>hello generic servlet</b>");
out.print("</body></html>");

}
}
```

HttpServlet class

- HttpServelt is an abstract class, it comes under package '**javax.servlet.http.HttpServlet**' .

- To create a servlet the class must extend the HttpServlet class and override at least one of its methods (doGet, doPost, doDelete, doPut).

- The HttpServlet class extends the GenericServlet class and implements a Serializable interface.

- HttpServlet is a dependent protocol and is only used with HTTP protocol.

Methods of HttpServlet Class

doGet() Method

- This method is used to handle the GET request on the server-side.

- This method also automatically supports HTTP HEAD (HEAD request is a GET request which returns nobody in response) request.

- The GET type request is usually used to *preprocess* a request.

Synatx:

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

Methods of HttpServlet Class

doPost() Method

- This method is used to handle the POST request on the server-side.

- This method allows the client to send data of unlimited length to the webserver at a time.

- The POST type request is usually used to post-process a request.

- **Synatx:**

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

Methods of HttpServlet Class

doHead() Method

- This method is overridden to handle the HEAD request.

- In this method, the response contains the only header but does not contain the message body.

- This method is used to improve performance (avoid computing response body).

- Synatx:

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

Methods of HttpServlet Class

doPut() Method

- This method is overridden to handle the PUT request.

- This method allows the client to store the information on the server(to save the image file on the server).

- This method is called by the server (via the service method) to handle a PUT request.

Synatx:

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

Other useful methods of Http Servlet

- **doDelete() Method**
  - allows a client to remove a document or Web page from the server.

- **doOptions() Method**
  - to determine which HTTP methods the server supports and returns an appropriate header.

- **service() Method**
  - receives standard HTTP requests from the public *service* method and dispatches them to the *doXXX* methods defined in this class.

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

How to use HttpServlet

```java
import javax.servlet.*;

import javax.servlet.http.*;

public class MyServlet extends HttpServlet {
    // we are defining the doGet method of HttpServlet
    // abstract class
    public void doGet(HttpServletRequest rq,HttpServletResponse rs)
    {
        // here user write code to handle doGet request
    }

    // we are defining the doPost method of HttpServlet
    // abstract class
    public void doPost(HttpServletRequest rq,HttpServletResponse rs)
    {
        // here user write code to handle doPost request

    }

} // class ends
```

GenericServlet	HttpServlet
It is defined by javax.servlet package.	It is defined by javax.servlethttp package.
It describes protocol-independent servlet	It describes protocol-dependent servlet.
GenericServiet is not dependent on any particular protocol. It can be used with any protocol such as HTTP, SMTP, FTP, and so on.	HttpServlet is a dependent protocol and is only used with HTTP protocol.
All methods are concrete except the service() method. service() method is an abstract method.	All methods are concrete (non-abstract). service() is non-abstract method. service() can be replaced by doGet() or doPost() methods.
The service method is abstract.	The service method is non-abstract
It forwards and includes a request and is also possible to redirect a request.	It forwards and includes a request but it is not possible to redirect the request.
GenericServlet doesn't allow session management with cookies and HTTP sessions.	HTTPServlet allows session management with cookies and HTTP sessions.
It is an immediate child class of Servlet interface.	It is an immediate child class of GenericServlet class.
GenericServlet is a superclass of HttpServlet class.	HttpServlet is a subclass of GenericServlet class.

Example 1: Servlet (submit & action)

Testing.html

<!DOCTYPE html>

<html>

<head>

<meta charset="*UTF-8*">

<title>My First Servlet</title>

</head>

<body>

<form action="*Testing1*">

<input type="*submit*" *value*="*Check Submit*">

</form>

</body>

</html>
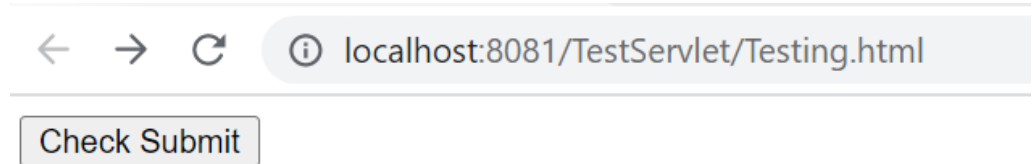
Example 1: Servlet

```java
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

public class Testing1 extends HttpServlet {

    // service method
    public void service(ServletRequest req, ServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();
        pw.println("<h2>Hello UPES</h2>");
        System.out.println("in service");
    }

}
```

← → C   ⓘ localhost:8081/TestServlet/Testing1?

Hello UPES

Example 2: Servlet (Text inputs)

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

<form action="ServletDemo">
Enter First number: <input type="text"
name="num1"><br><br>
Enter Second Number:<input type="text"
name="num2"><br><br>
<input type="submit">
</form>

</body>
</html>
```
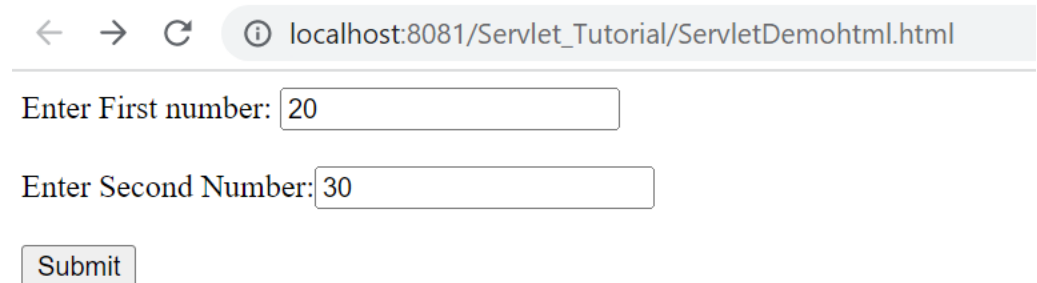
← → C ⓘ localhost:8081/Servlet_Tutorial/ServletDemohtml.html

Enter First number: `20`

Enter Second Number:`30`

Submit

Deepak Sharma, Asst. Professor UPES Dehradun

Default method: get()

Example 2: Servlet (Text inputs)

```java
public class ServletDemo extends HttpServlet {

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

int a=Integer.parseInt(request.getParameter("num1"));
int b=Integer.parseInt(request.getParameter("num2"));
int s=a+b;
PrintWriter out= response.getWriter();
out.println("Sum of "+a+" and "+b+" is "+s);

}
```

← → C ⓘ localhost:8081/Servlet_Tutorial/ServletDemo?num1=20&num2=30

Sum of 20 and 30 is 50

Note: Check URL-> Get method showing values

Deepak Sharma, Asst. Professor UPES Dehradun

Example 3: Servlet -Post() method

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

<form action="ServletDemo" method="post">
Enter First number: <input type="text" name="num1"><br><br>
Enter Second Number:<input type="text" name="num2"><br><br>
<input type="submit">
</form>

</body>
</html>
```
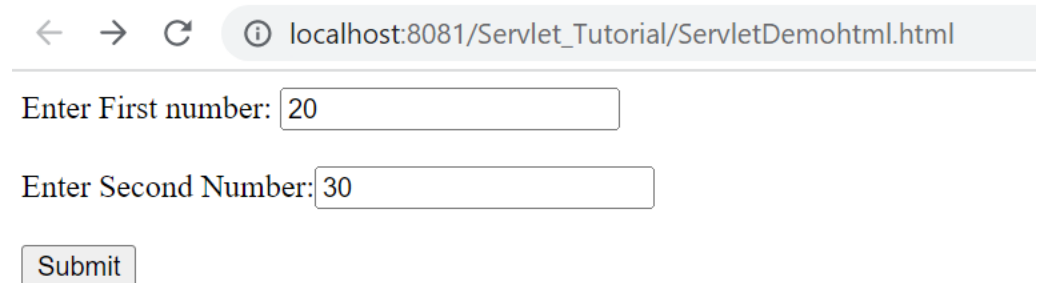
← → C   ⓘ localhost:8081/Servlet_Tutorial/ServletDemohtml.html

Enter First number: `20`

Enter Second Number:`30`

Submit

Example 3: Servlet -doPost() method

```java
public class ServletDemo extends HttpServlet {


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

int a=Integer.parseInt(request.getParameter("num1"));
int b=Integer.parseInt(request.getParameter("num2"));

int s=a+b;

PrintWriter out= response.getWriter();
out.println("Sum of "+a+" and "+b+" is "+s);
}

}
```

← → C ⓘ localhost:8081/Servlet_Tutorial/ServletDemo

Sum of 20 and 30 is 50

Note: Values are not visible in URL

Example 4: Servlet (Print current Date and Time)

```java
public class ServletDemo extends HttpServlet {

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

 response.setContentType("text/html");
 PrintWriter pw = response.getWriter();
 java.util.Date date = new java.util.Date();
 pw.println("<h2>"+"Current Date & Time: " +date.toString()+"</h2>");
 pw.close();
}
}
```

← → C ⓘ localhost:8081/Servlet_Tutorial/ServletDemo

Current Date & Time: Tue Nov 29 11:10:02 IST 2022

JSP – Java Server Pages

- JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

- A JSP page consists of HTML tags and JSP tags.

- The JSP pages are easier to maintain than Servlet because we can separate designing and development.

- It provides some additional features such as Expression Language, Custom Tags, etc.

Advantages of JSP over Servlet

1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.
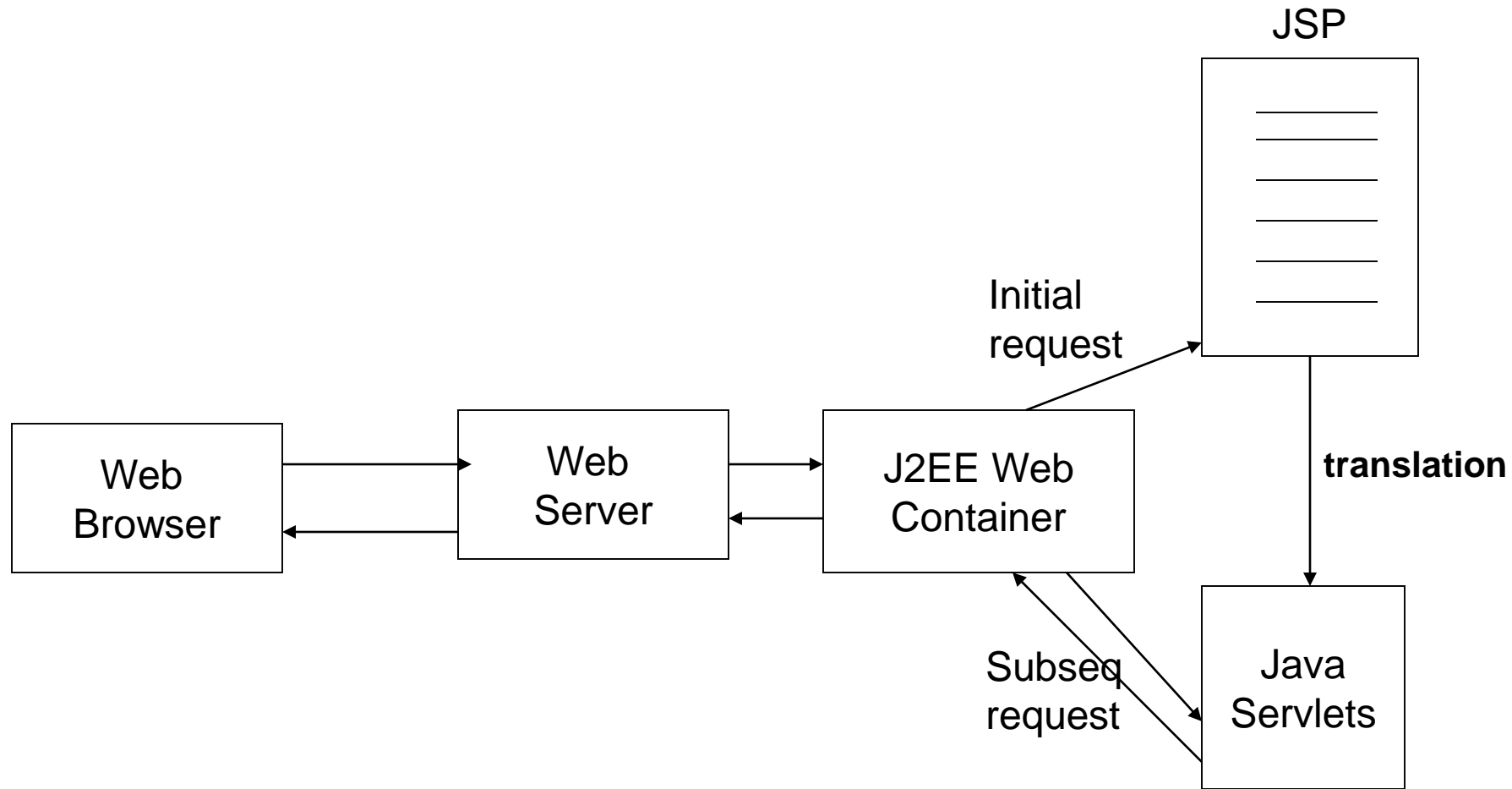
3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.
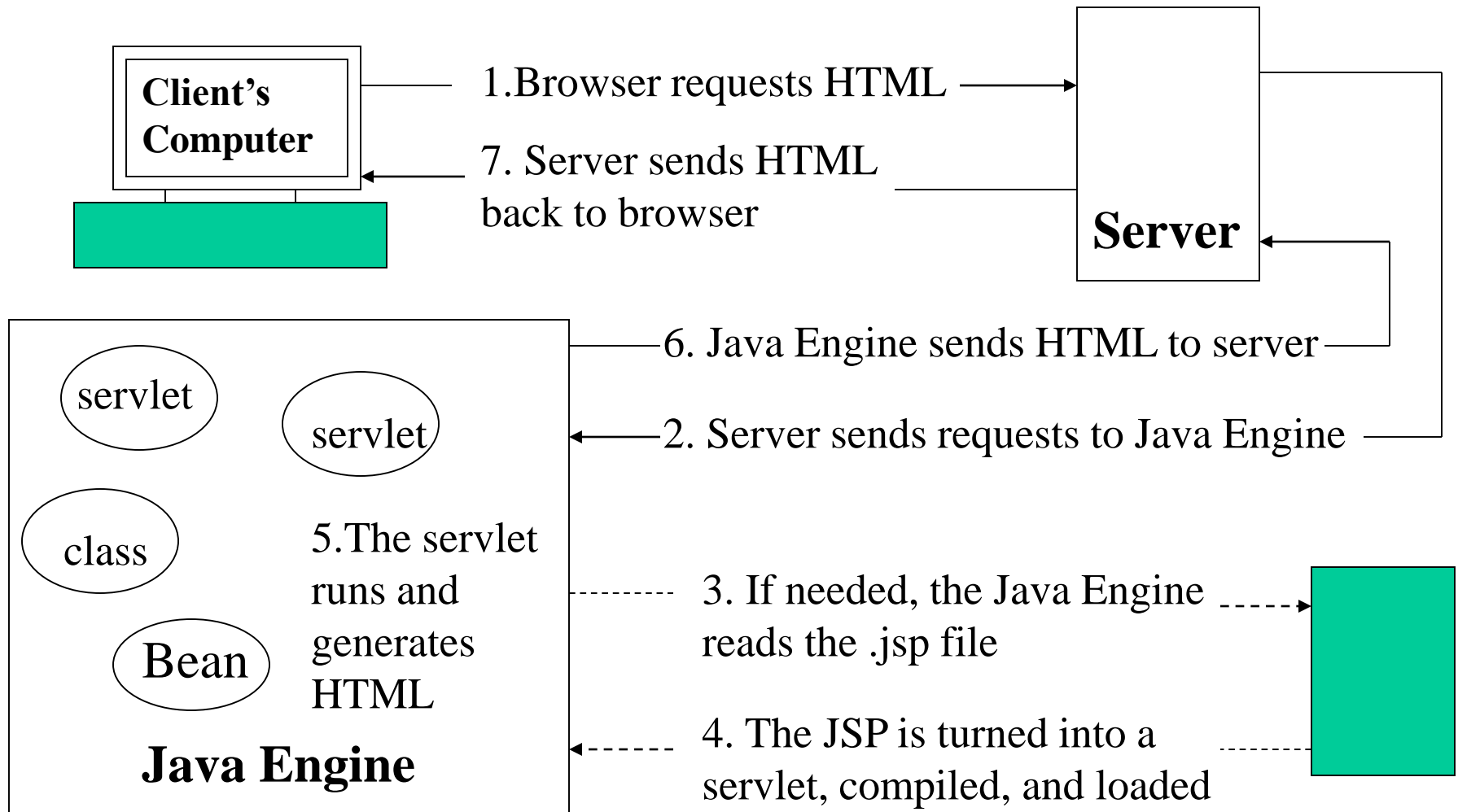
4) Less code than Servlet

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

JSP compilation into Servlets

JSP

Initial
request

Web
Browser

Web
Server

J2EE Web
Container

translation

Subseq
request

Java
Servlets

Java Server Pages (JSP)

Client's Computer

1. Browser requests HTML →

Server

7. Server sends HTML back to browser

6. Java Engine sends HTML to server

2. Server sends requests to Java Engine

Java Engine

- servlet
- servlet
- class
- Bean

5. The servlet runs and generates HTML

3. If needed, the Java Engine reads the .jsp file

4. The JSP is turned into a servlet, compiled, and loaded

The Lifecycle of a JSP Page

The JSP pages follow these phases:

- Translation of JSP Page

- Compilation of JSP Page

- Classloading (the classloader loads class file)

- Instantiation (Object of the Generated Servlet is created).

- Initialization (the container invokes jspInit() method).

- Request processing (the container invokes _jspService() method).

- Destroy (the container invokes jspDestroy() method).
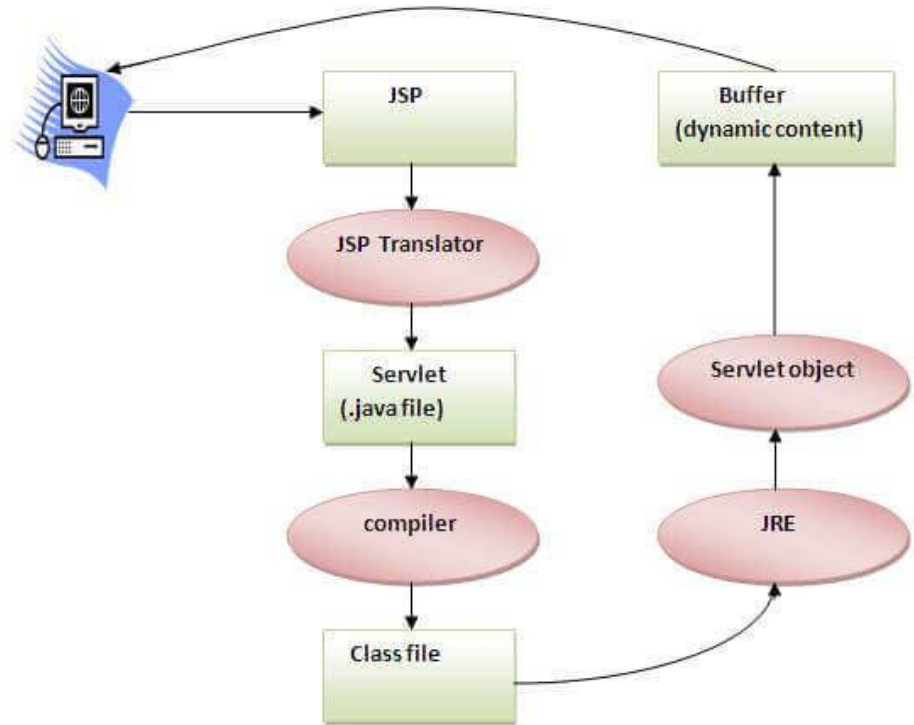
Note: jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP.

Creating a simple JSP Page

index.jsp

```
<html>
<body>
<% out.print(2*5); %>
</body>
</html>
```

← → C ⓘ localhost:8081/TestServlet/index.jsp

10

Example1 : JSP

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>My First Servlet</title>
</head>
<body>

<p>Enter two numbers and click the 'calculate' button.</p>
<form action="calculator.jsp" method="get">
<input type=text name=value1><br>
<input type=text name=value2 ><br>
<input type=submit name=calculate value=calculate>

</form>
</body>
</html>
```
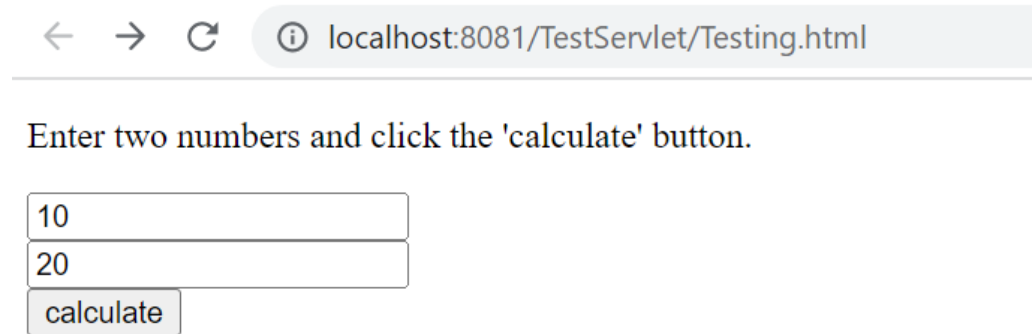
← → C　　① localhost:8081/TestServlet/Testing.html

Enter two numbers and click the 'calculate' button.

10

20

calculate

Example 1: JSP

calculator.jsp

```jsp
<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head><title>A simple calculator: results</title></head>
<body>
<%-- A simpler example 1+1=2 --%>
1+1 = <%= 1+1 %>
<%-- A simple calculator --%>
<h2>The sum of your two numbers is:</h2>
<%= Integer.parseInt(request.getParameter("value1")) +
    Integer.parseInt(request.getParameter("value2")) %>
</body>
</html>
```

← → C ⓘ localhost:8081/TestServlet/calculator.jsp?value1=10&value2=20&calculate=calculate

1+1 = 2

The sum of your two numbers is:

30

JSP Tags

- Comments   <%-- …...text…... --%>
- Declaration   <% ! int i; %>

       <% ! int numOfStudents(arg1,..) { } %>

- Expression   <%= 1+1 %>
- Scriptlets   <% … java code … %>
- include file   <%@ include file="*.jsp" %>
- …...

JSP Scripting elements

- The scripting elements provides the ability to insert java code inside the jsp.

- scriptlet tag

- expression tag

- declaration tag

JSP scriptlet tag

- A scriptlet tag is used to execute java source code in JSP.

<% java source code %>

```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>
</html>
```

Example of JSP scriptlet tag that prints the user name

File: index.html

```html
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

File: welcome.jsp

```jsp
<html>
<body>
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
</form>
</body>
</html>
```

JSP expression tag

- The code placed within **JSP expression tag** is *written to the output stream of the response*. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

```
<%= statement %>
```

```
<html>
<body>
<%= "welcome to jsp" %>
</body>
</html>
```

Note: Do not end your statement with semicolon in case of expression tag.

Example of JSP expression tag that prints current date and time

```
<html>
<body>
<%
 out.print(new java.util.Date());
%>
</body>
</html>
```

Example of JSP expression tag that prints the user name

File: index.jsp

```html
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname"><br/>
<input type="submit" value="go">
</form>
</body>
</html>
```

File: welcome.jsp

```jsp
<html>
<body>
<%= "Welcome "+request.getParameter("uname") %>
</body>
</html>
```

JSP Declaration Tag

- The **JSP declaration tag** is used *to declare fields and methods.*

<%! field or method declaration %**>**

Example of JSP declaration tag that declares field

index.jsp

```
<html>
<body>
<%! int data=50; %>
<%= "Value of the variable is:"+data %>
</body>
</html>
```

Example of JSP declaration tag that declares method

index.jsp

```jsp
<html>
<body>
<%!
int cube(int n){
return n*n*n*;
}
%>
<%= "Cube of 3 is:"+cube(3) %>
</body>
</html>
```