Servlets

Introduction

Agenda

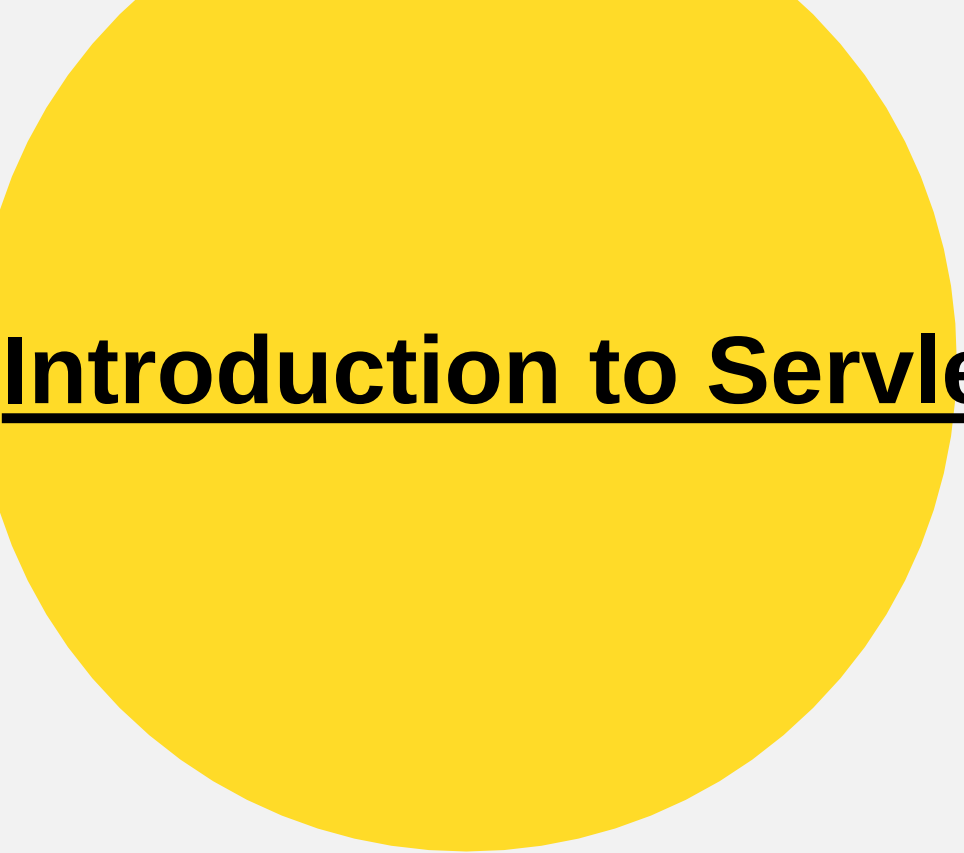
1. Introduction to Servlets
2. Deploying a Simple Servlet
3. Servlet Life Cycle

Objectives

At the end of this module, you will be able to:

- Describe the role of HTTP Servlet in Web Programming
- Describe and use the Servlet Life Cycle methods appropriately

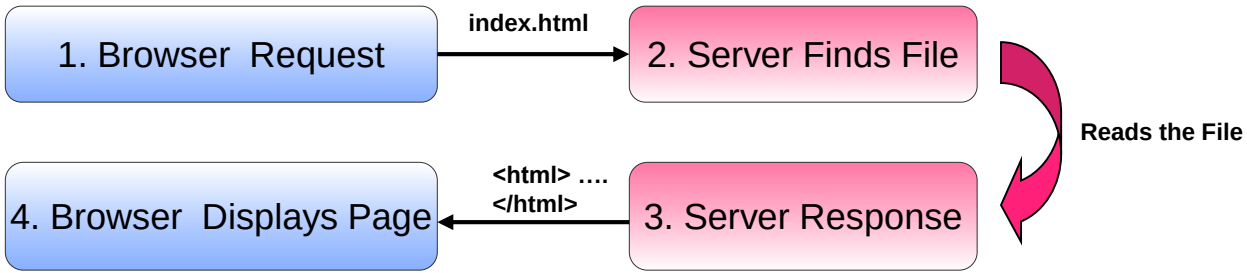
Introduction to Servlets





Server-side Programming

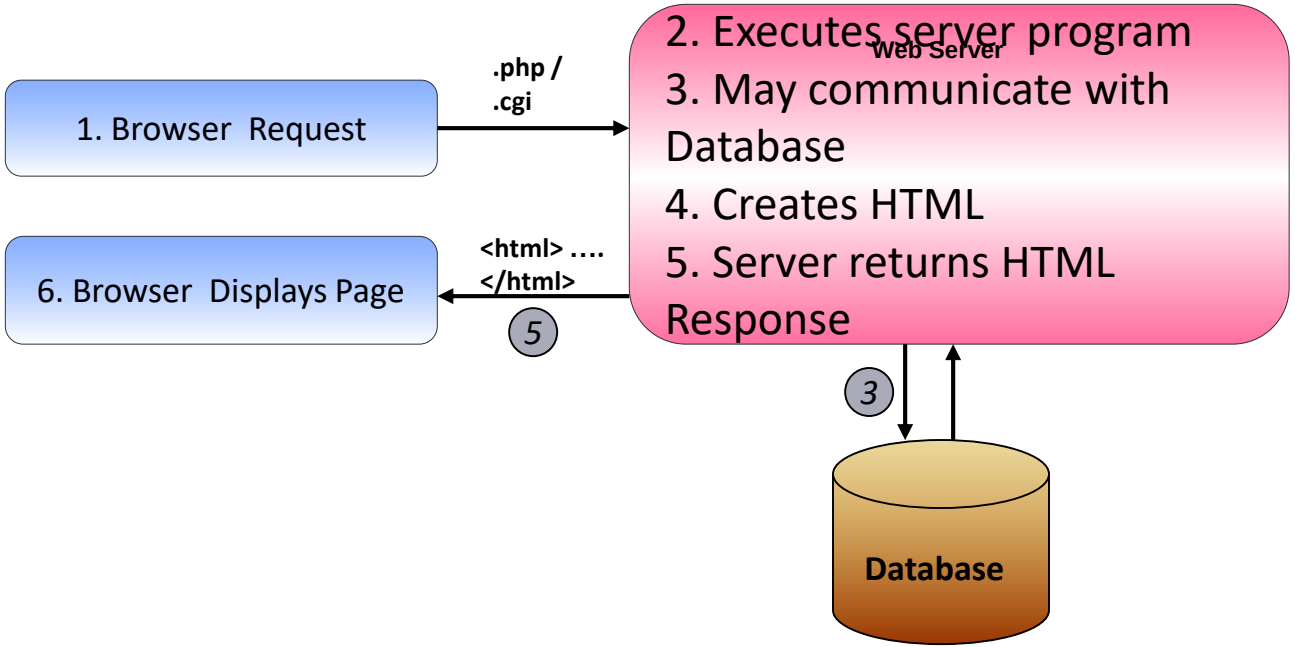
- Static HTTP transaction - Browser requests for index.html, and Server responds with HTML file



HTTP (Hyper Text Transport Protocol) is the protocol that clients and servers use on the web to communicate

Server-side Programming (Contd.).

- Dynamic HTTP transaction - Browser requests OrderServlet.class, server runs program that creates HTML, server returns HTML to browser

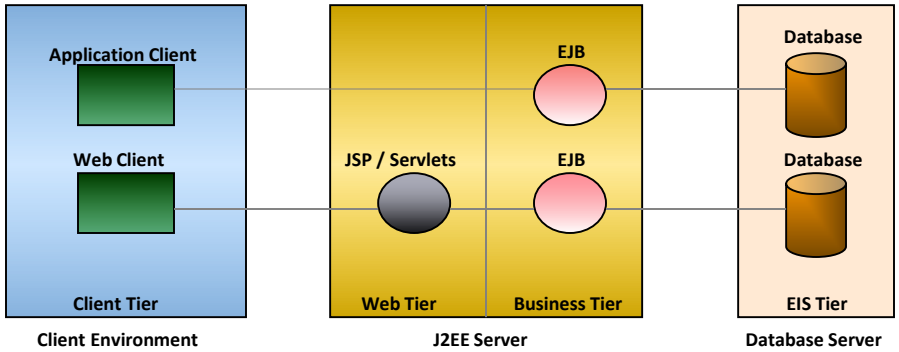


Web Server

- A computer having server software installed within it which serves up web pages
- A program that uses client/ server model and World Wide Web's Hypertext Transfer Protocol (HTTP)
- Responsible for accepting HTTP requests from clients (web browsers) and serving HTTP responses which are web pages such as HTML documents
- Popular web servers
  - Apache HTTP Server (Apache)
  - Microsoft Internet Information Server (IIS)
  - Sun Java System Web Server

Java server-side web components

- **A web component is a software entity that runs on a web server**
  - Provides it with the capabilities needed for dynamically handling client requests and generating web presentation content
- **The J2EE specification defines two types of web components**
  - Servlets
  - Java Server Pages(JSPs)



J2EE Application N-Tiered Architecture

What are Servlets?

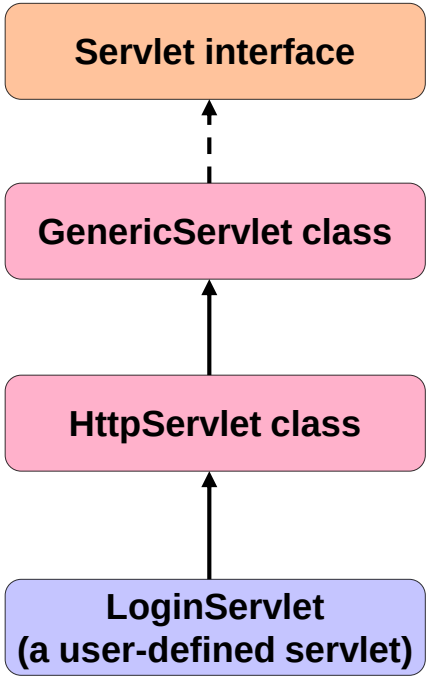
- A Java class that runs on a web server and dynamically handles client requests
- It extends the functionality of a web server by receiving client requests and dynamically generating a response
- Since servlets are Java-based, they are platform independent

Uses of Servlets

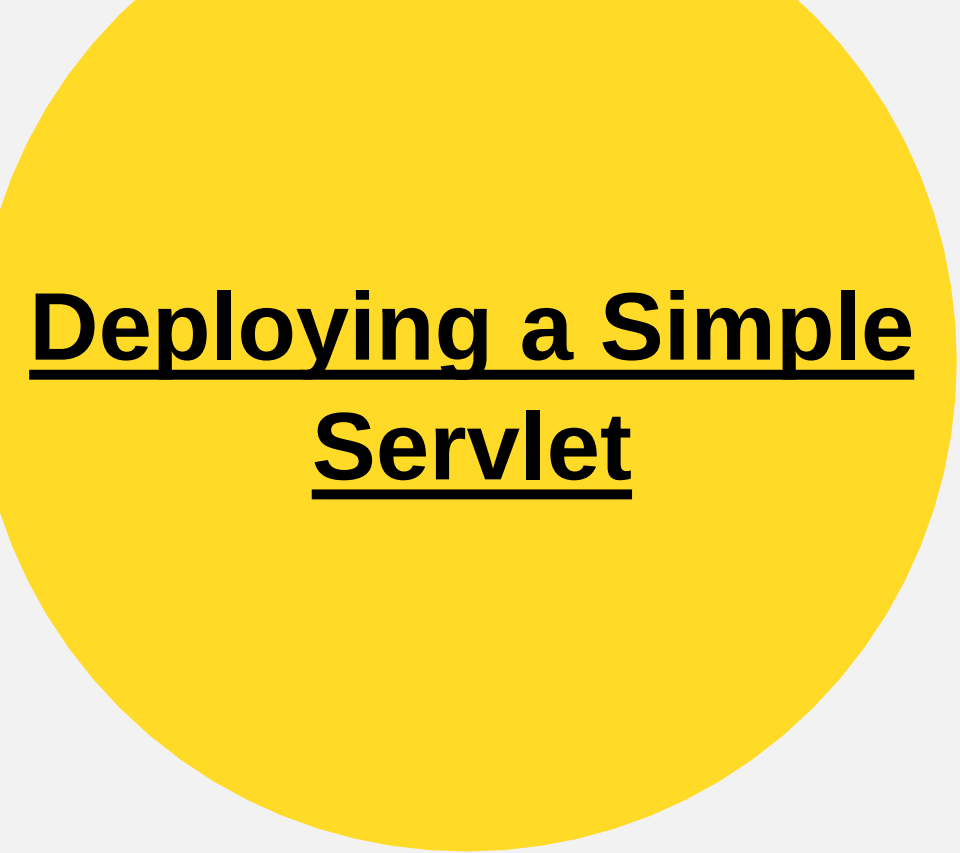
- Processing and/or storing data submitted by an HTML form
  - Example: Processing data of a login form
- Providing dynamic content
  - Example: Returning results of a database query to the client
- Managing state information on top of the stateless HTTP
  - Example: For an online shopping cart system which manages shopping carts for many concurrent customers and maps every request to the right customer

Servlet Architecture Overview

- Java Servlet API are in packages javax.servlet and javax.servlet.http
  - Provide interfaces and classes for writing servlets
- All servlets must implement Servlet interface
  - Defines life-cycle methods
- Extend GenericServlet class
  - To implement generic services
- Extend HttpServlet class
  - To implement HTTP-specific services



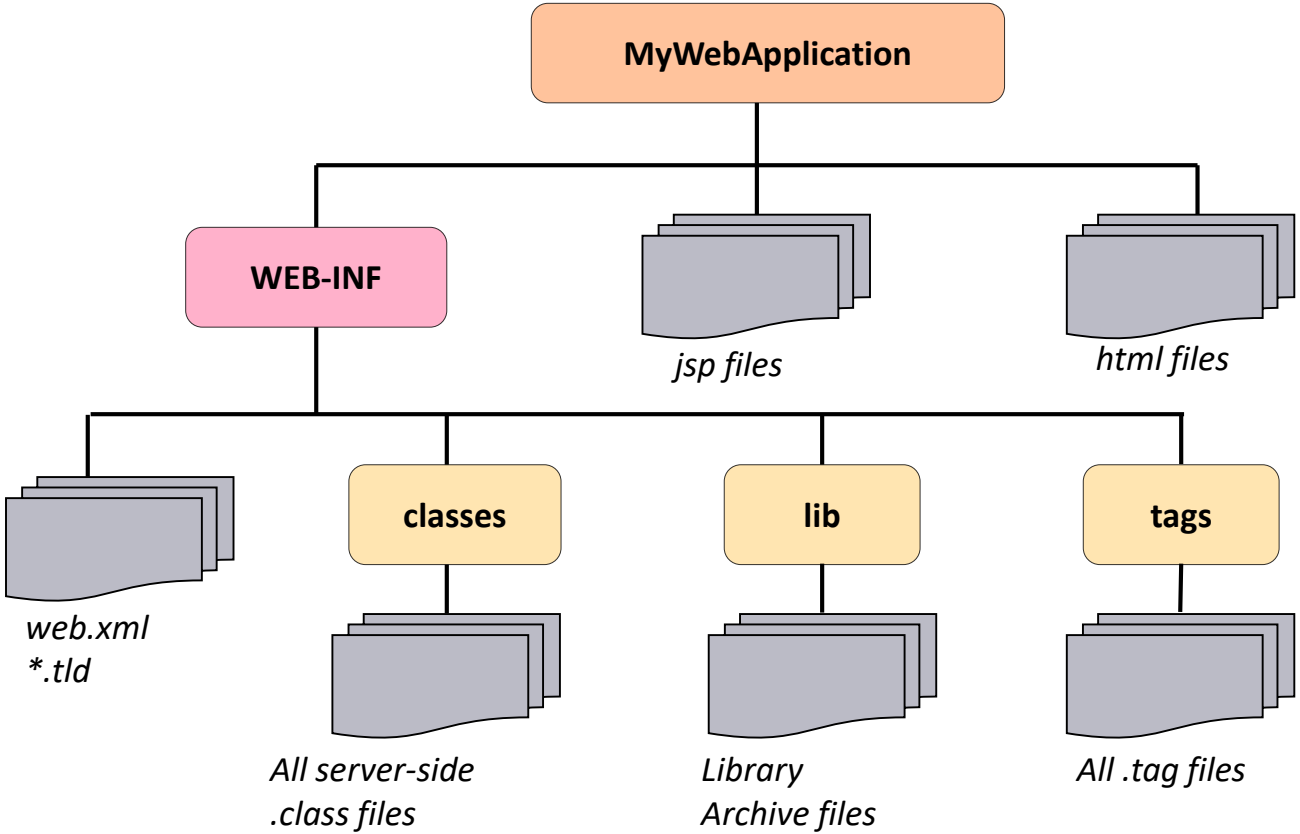
Deploying a Simple Servlet



Steps for writing and deploying simple servlet

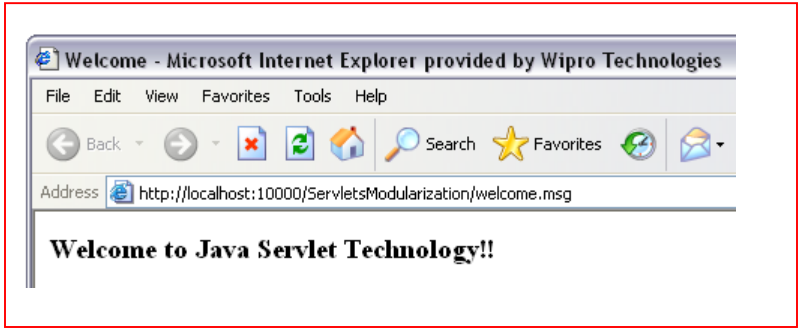
1. Set environment variables: JAVA_HOME, CATALINA_HOME, CLASSPATH, and PATH
2. Create a directory, say "MyWebApplication" in **C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps**
3. Compile your servlet say "WelcomeServlet.java" in **C:\Program Files\Apache Software Foundation\Tomcat5.0\webapps\MyWebApp\WEB-INF\classes** directory
4. Create web.xml file and place it in **C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\MyWebApplication\WEB-INF** folder
5. Start Tomcat server by double-clicking **C:\Program Files\Apache Software Foundation\Tomcat 5.0\bin\startup.batch** file OR by selecting Start -> Programs -> Apache Tomcat 5.0 -> Monitor Tomcat
6. Open browser with URL: http://localhost:10000/ to check Tomcat is successfully installed
7. Give URL as http://localhost:10000/MyWebApplication/WelcomeServlet to test your servlet

Directory Structure of a Web application



Demo for a Simple Servlet

- A simple HTTP Servlet that displays a Welcome message on the web page



- Files required:
  - WelcomeServlet.java
  - web.xml

Demo for a Simple Servlet (Contd.).

- An HTTP Servlet that displays a Welcome message

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

public class WelcomeServlet extends HttpServlet {
    public void doGet(HttpServletRequest req,HttpServletResponse res)
                    throws ServletException, IOException {
        res.setContentType("text/html");   // set header field first
        PrintWriter pw = res.getWriter(); // then get writer & write response data
        pw.println("<HTML>");
        pw.println("<HEAD><TITLE>Welcome</TITLE></HEAD>");
        pw.println("<BODY>");
        pw.println("<H3>" + "Welcome to Java Servlet Technology!!" + "</H3>");
        pw.println("</BODY>");
        pw.println("</HTML>");
        pw.close();   //closes the writer
    }
}
```

The Web Deployment Descriptor – web.xml

- An XML file web.xml is a deployment descriptor that describes
  - mapping from URIs to application resources
  - initialization parameters
  - security constraints
  - registration of listeners and filters
- Example: web.xml

```
<web-app>
        <display-name>A Small Web Application</display-name>
        <servlet>
                <servlet-name>MyFirstServlet</servlet-name>
                <servlet-class>WelcomeServlet</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>MyFirstServlet</servlet-name>
                <url-pattern>/welcome.msg</url-pattern>
        </servlet-mapping>
</web-app>
```

Web Container

Web components and their container run on J2EE server

- Provides execution environment for servlets and JSPs of a web application
- Manages execution of JSP and servlet components for J2EE applications

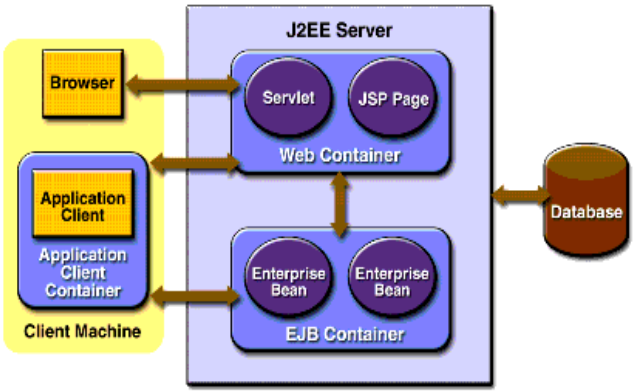


Fig: J2EE Server and Container Types

Role of a Web Container

- Communication Support
  - Provides an easy way for servlets to talk to web server
- Lifecycle Management
  - Controls lifecycle of servlets
- Multithreading Support
  - Automatically creates a new Java thread for every servlet request it receives
- Declarative Security
  - Enables to configure security in an XML deployment descriptor thereby avoiding hard-coding it in servlet or any other class code
- JSP Support
  - Does JSP processing

How web container handles Servlet requests

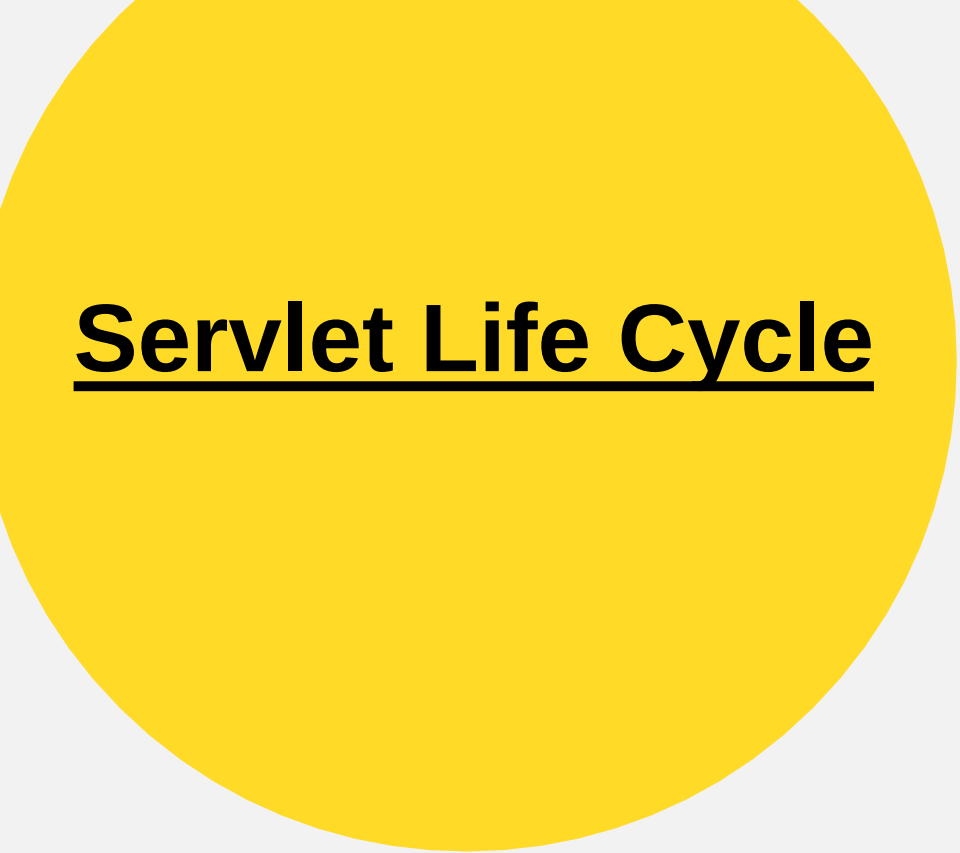
- Container creates 2 objects on receiving a request for a servlet: HttpServletRequest and HttpServletResponse
- Finds right servlet based on URL in the request
- Creates a thread for that request
- Passes request and response objects to the servlet thread
- Calls servlet's service() method
  - The service() method in turn calls doGet or doPost based on type of request (Assume request was an HTTP GET)
  - The doGet method generates dynamic page and captures it in response object
- Converts response object into an HTTP response on completion of thread
- Sends this response to the client
- Finally deletes the request and response objects

Knowledge Checkpoint

1. Suppose you are a web developer working for an Online Movie Service. You want to use a servlet called MovieServlet so that clients can access the latest film shows for the day in a particular city from your movie database. Determine the correct sequence of following steps carried out by MovieServlet when processing a request from a client

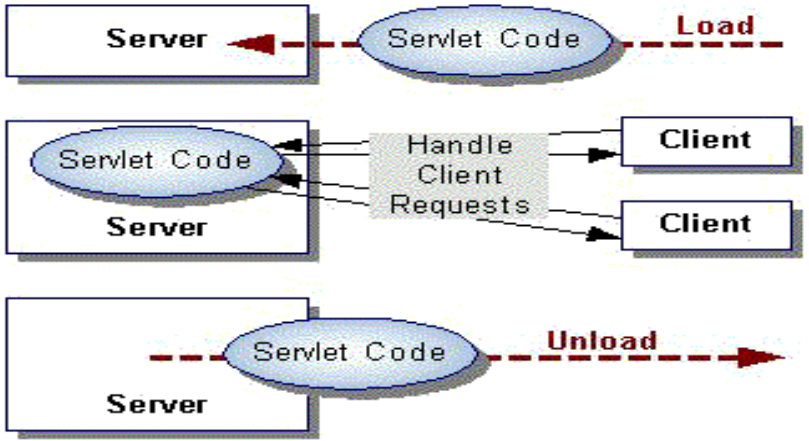
Sl.No	Description
1	Check information included in the Http request
2	Access any necessary business components or data storage
3	Set the appropriate Http response parameters
4	Read data submitted by the client
5	Send the response to the client
6	Format the results in a response

Servlet Life Cycle



Life Cycle of a Servlet

- An HTTP servlet's life cycle is controlled by the web container where it is deployed



Source: http://www.iam.ubc.ca/guides/javatut99/servlets/lifecycle/index.html

Servlet interface

- Provides methods that manage the servlet and its communications with clients
  - **init(ServletConfig)**
    Initializes the servlet. Runs once before any requests can be serviced
  - **service(ServletRequest, ServletResponse)**
    Processes a single request from the client
  - **destroy()**
    This method releases all the resources
  - **getServletConfig()**
    Returns a servlet config object and this object contains any initialization parameters and startup configuration information for this servlet
  - **getServletInfo()**
    Returns a string containing information about the servlet, such as its author, version, and copyright

Lifecycle Methods

- Interaction between a web server, a servlet, and a client is controlled using the life-cycle methods
- A servlet's life cycle methods are
  - init()
  - service()
  - destroy()
- The init() and destroy() methods will be called *only once* during the life time of your Servlet
- The service() and it's broken down methods (doGet(), doPost() etc) will be called as many times as requests are received for them by the web container

Initializing a servlet

- Web container initializes servlet after web container loads and instantiates servlet class and before it delivers requests from clients

- The init method is invoked only once during servlet's lifetime – when servlet is first created

- Override init method of Servlet interface if you want the servlet to
  - Read persistent configuration data
  - Initialize resources
  - Perform any other one-time activities

- Two versions of init method – one that takes no arguments and one that takes a ServletConfig object as an argument
  - init( ) - use this when your servlet does not need any specific initialization
  - init(ServletConfig) - use this when your servlet needs to check specific settings before completing initialization

Servicing client requests

- Once servlet is loaded and initialized, the servlet is able to handle client requests
- Web container processes the requests in servlet's service method
- Every time the server receives an incoming request for a servlet, it generates a new thread and calls the service method
- The service method then checks the HTTP request type and calls the appropriate doXXX method
- Syntax: public void service(ServletRequest req, ServletResponse res)
- Role of service method
  - To extract information from the request
  - Access external resources
  - Populate the response based on that information

Destroying a Servlet

- Server may unload a servlet instance
  - If the servlet has been idle for some time (default is 30 minutes) or
  - If the web application is undeployed or
  - If server shuts down

- Before it removes the servlet, it calls destroy method only once

- Before servlets get destroyed, this method helps in
  - Cleanup operations such as closing database connections
  - Halting background threads
  - Releasing other resources such as IO streams

- Syntax: public void destroy()

Summary

In this module, you were able to:

- Describe the role of HTTP Servlet in Web Programming
- Describe and use the Servlet Life Cycle methods appropriately

Quiz

1. The doGet() or doPost() method of a Servlet are invoked by ----------
   a) init() method
   b) service() method
   c) destroy() method

- ---------- is the deployment descriptor file for Servlets
  a) servlet-config.xml
  b) web.xml
  c) struts-config.xml

References

1. TutorialsPoint.COM (2012). *Servlets Tutorial*. Retrieved April 1, 2012 from http://www.tutorialspoint.com/servlets/index.htm

2. Stephanie Bodoff. *Oracle Sun Developer Network: JavaServer Pages Technology*. Retrieved April 1, 2012, from, http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets.html

3. SkillSoft (2003).*Developing a basic HTTP servlet*. Retrieved April 10, 2012, from, http://www.alc.amadeus.com/content/public/alw/skillsoft/cbtlib/73468/73473/eng/thin/transcript.html

4. Saravanan Sivaji (2009). *Developing a Basic Servlet Program*. Retrieved April 10, 2012, from,http://saravananmtech.wordpress.co m/2009/04/

5. Javatut99 (1999). *Lifecycle of a Servlet*. Retrieved April 10, 2012, from, http://www.iam.ubc.ca/guides/javatut99/servlets/lifecycle/index.html

Thank You