HNDIT4232-
Enterprise
Architecture

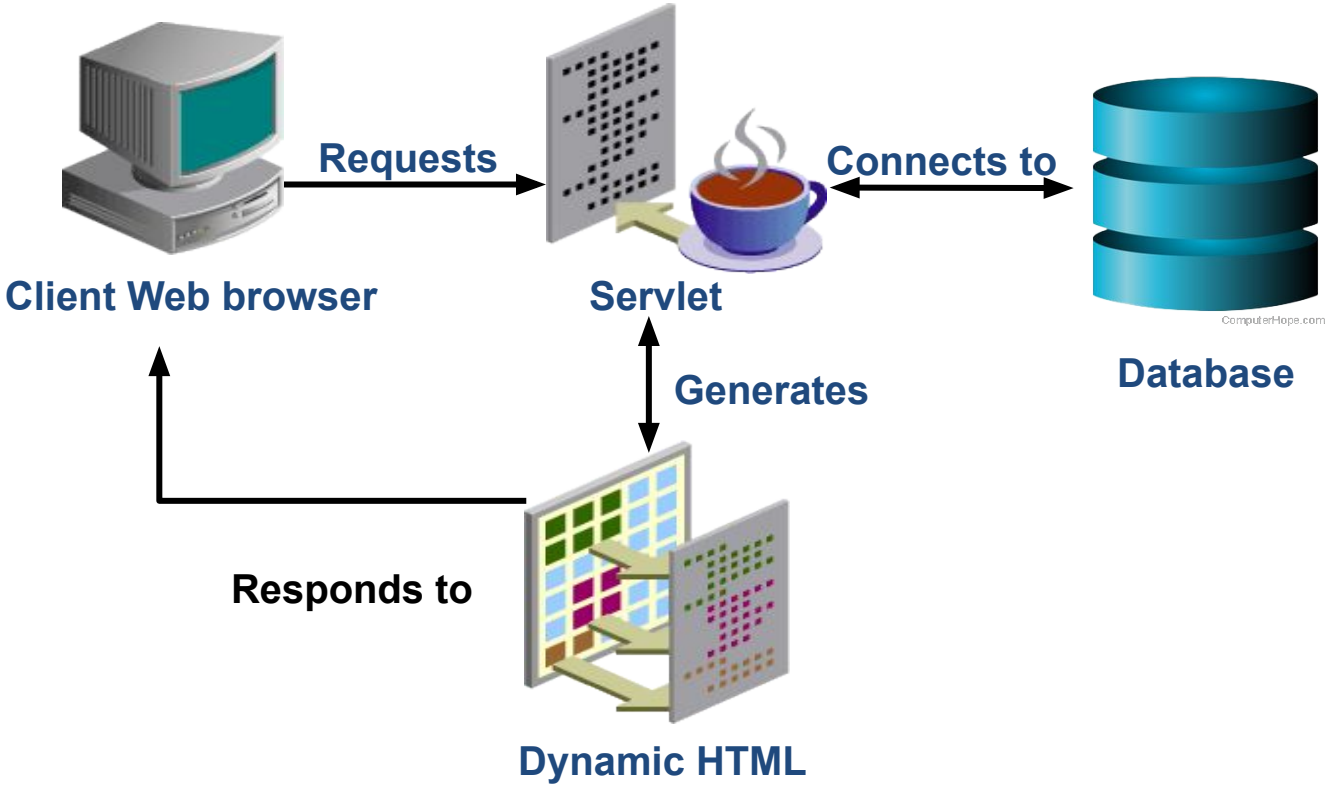
Creating the Web Tier: Java Servlets

Bandula Kularatne

- Overview
- How Browser and Web Sever Communicates
- What is a Servlet
- Features
- Java Servlets and JSP
- Life Cycle of Servlets
- HTTP POST vs. GET methods

Overview – Java Servlets

Java Servlets are Java-based server-side components that dynamically generate content and handle client requests in web applications. They are part of the Java Enterprise Edition (Java EE)



What is a Servlet?

- A Java Servlet is server side program that called by user interface or by another web component and contain business logic to process a request
- Servlets handle client requests
- Servlets serve the same purpose as same as the program implementing with CGI. But servlet offer some advantages over CGI.
- Although not exclusively, most servlets are used to answer HTTP requests and hence extend the

  javax.servlet.http.HttpServlet class

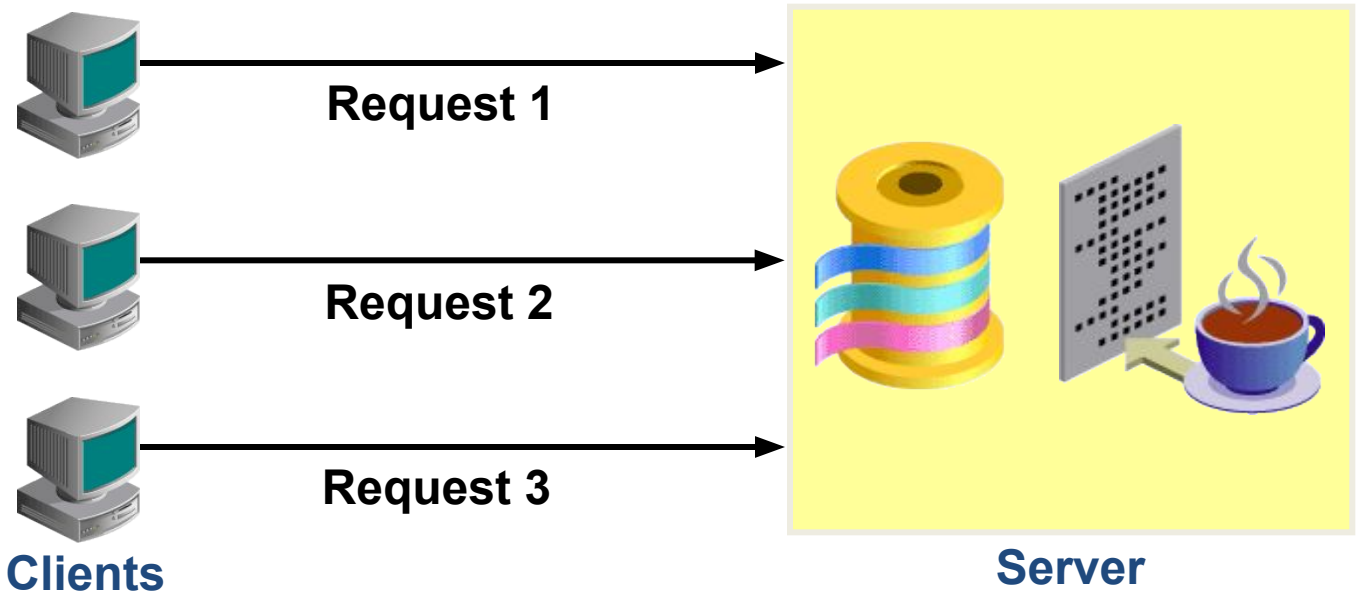
Browser and Web Sever

- The user enters a Uniform Resource Locator(URL)
- Browser generate a HTTP request
- Server redirect to the request to the index file of the specified web site
- web server constructs a dynamic web page by creating separate process
- It is communicate with the server using an interface –

  Ex: Java Servlets, Common Gateway Interface (CGI) etc…
- Returns a HTTP response to the client.

Features of Servlets

- Concurrent requests are possible and common.
- Servlet methods are run in threads.
- Servlet instances are shared by multiple client requests.

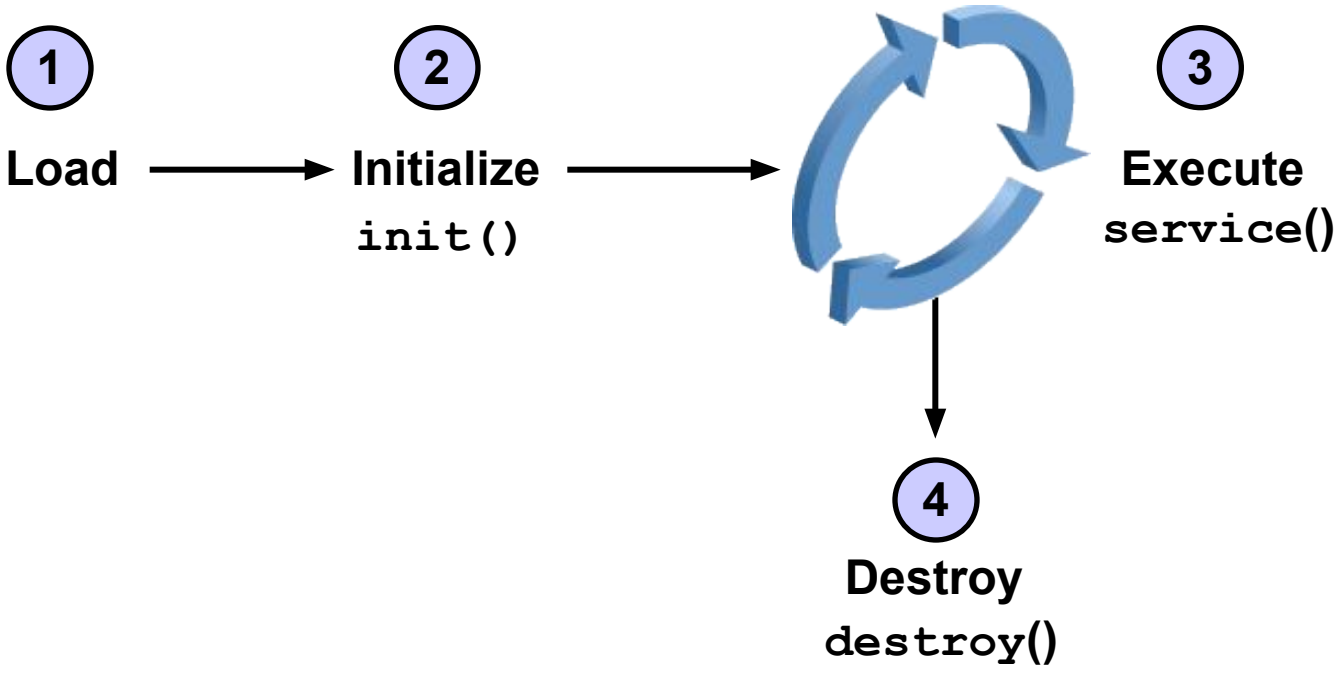


Java Servlets and JSP

- Java offers two ways to make web applications
  - Java Servlets
    - Java code with HTML inside
  - JSP (JavaServer Pages)
    - HTML with Java code inside
- Servlets and JSP are often/best used in combination
- Servlets and JSP are part of Java Enterprise Edition
  - Features to be run by a web or application server

Life Cycle of Servlets

- All actions are carried out inside the server.
- After initial setup, the response time is less.



HTTP POST vs. GET methods

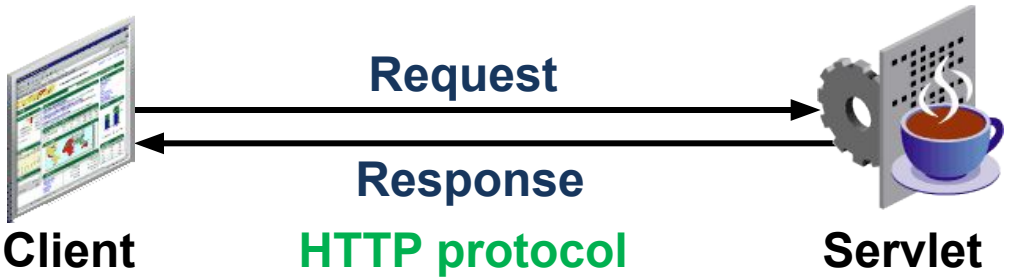
- The HTTP protocol offers two way to send data from a form to the server
  - POST
    - Data is carried in the body of the request
    - <form action="welcome.jsp" method="POST“ >
  - GET
    - Data is carried in the request URL
      - http://localhost:8084/firstwebapp/welcome.jsp?username=Anders&password=secret
    - Default in Java
    - Good for debugging
      - Easy to see data

Servlets are responsible for

- **Handling HTTP Requests:** Servlets receive HTTP requests from clients (such as web browsers) and process them accordingly.
- **Generating Dynamic Content:** Servlets generate dynamic content that is sent back to clients as HTTP responses. (HTML,XML,JSON…)
- **Processing Form Data:** Servlets handle form submissions by parsing request parameters and processing form data.
- **Managing Sessions:** Servlets manage user sessions to maintain stateful interactions with clients.
- **Accessing Databases:** Servlets interact with databases to retrieve, manipulate, and store data. Ex:JDBC
- **Implementing Business Logic:** Servlets encapsulate business logic and application-specific behavior.

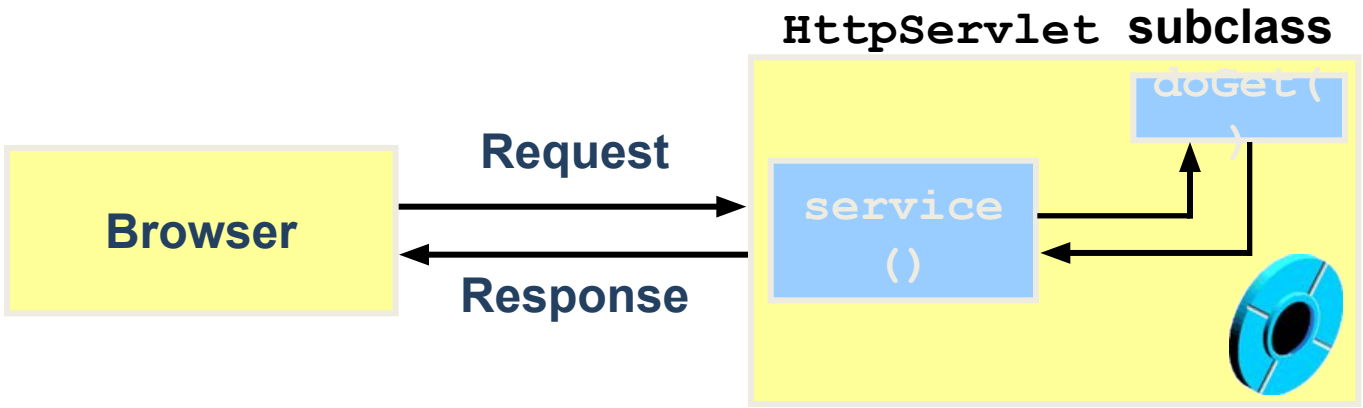
HTTP Servlets

- HTTP servlets extend the `HttpServlet` class, which implements the `Servlet` interface.
- A client makes an HTTP request, which includes a method type that:
  - Can be either a `GET` or `POST` method type
  - Determines what type of action the servlet will perform
- The servlet processes the request and sends back a status code and a response.



Inside an HTTP Servlet

- The servlet overrides the `doGet()` or the `doPost()` method of the `HttpServlet` class.
- The servlet engine calls the `service()` method, which in turn calls one of the appropriate `doXxx()` methods.
- These methods take two arguments as input:
  - `HttpServletRequest`
  - `HttpServletResponse`



Servlet: Example

```
• import javax.servlet.*;
• import javax.servlet.http.*;
• import java.io.*;
• public class SimplestServlet extends
  HttpServlet
• {
•   public void doGet(HttpServletRequest
  request, HttpServletResponse response)
  throws ServletException, IOException
•   {
•     PrintWriter out = response.getWriter();
•     out.println("Hello World");
•   }
• }
```

The `doGet()` Method

- The most common HTTP request method type made to a Web server is `GET`.
- The `service()` method in your servlet invokes the `doGet()` method. The `service()` method is invoked on your behalf by the Web server and the servlet engine.
- The `doGet()` method receives two parameters as input:
  - `HttpServletRequest`
  - `HttpServletResponse`
- Pass parameters by appending them to the URL `http://www.oracle.com/servlet?param1=value1`

The `doPost()` Method

- The `doPost()` method can be invoked on a servlet from an HTML form via the following:

```
<form method="post" action=…>
```

- The `service()` method in your servlet invokes the `doPost()` method. The `service()` method is invoked by the Web server and the servlet engine.
- The `doPost()` method receives two parameters as input:
  - `HttpServletRequest`
  - `HttpServletResponse`
- Pass parameters using the form field names

```
<input type="text" name="param1">
```

The `HttpServletRequest` Object

- The `HttpServletRequest` object encapsulates the following information about the client:
  - Servlet parameter names and values
  - The remote host name that made the request
  - The server name that received the request
  - Input stream data
- You invoke one of several methods to access the information:
  - `getParameter(String name)`
  - `getRemoteHost()`
  - `getServerName()`

The `HttpServletResponse` Object

- The `HttpServletResponse` object encapsulates information that the servlet has generated:
  - The content length of the reply
  - The MIME type of the reply
  - The output stream
- You invoke one of several methods to produce the information:
  - `setContentLength(int length)`
  - `setContentType(String type)`
  - `getWriter()`

Methods for Invoking Servlets

- Invoke servlets from a client by:
  - Typing the servlet URL in a browser
  - Embedding the servlet URL in an HTML or a JavaServer Page (JSP) page, or another servlet (an `href` link)
  - Submitting a form to the servlet (via the `action` tag)
  - Using URL classes in client Java applications
- Invoke servlets inside the J2EE container by defining a chain of servlets or JSPs.

Your First Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
  public void doGet(
        HttpServletRequest  request,
        HttpServletResponse response)
  throws ServletException, IOException{
    response.setContentType ("text/html");
       PrintWriter out =
response.getWriter();
    out.println ("<html>");
       out.println ("<body>");
    out.println ("Hello World!");
    out.println ("</body></html>");
  }
}
```

Handling Input: The Form

- You can use an HTML form and the `doPost()` method to modify the `HelloWorld` servlet.

```
<html><body>
<form method="post" action="newhelloworld">
Please enter your name. Thank you.
<input type="text" name="firstName"> <P>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Handling Input: The Servlet

```
public class NewHelloWorld extends HttpServlet {
  public void doPost(
  HttpServletRequest  req, HttpServletResponse res)
  throws ServletException, IOException{
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println ("<html><body>");
    String name = req.getParameter("firstName");
    if ((name != null) && (name.length() > 0))
      out.println ("Hello: " + name +
      "  How are you?");
    else
      out.println ("Hello Anonymous!");
    out.println ("</body></html>");
  }}
```

Summary

- In this lesson, you should have learned how to:
  - Java Servlet
  - Describe the servlet life cycle
  - Develop and run a servlet
  - Collect information from a client
  - Respond to the client
  - Deploy a servlet to Apache Tomcat Server