

# Web Technologies

*HTML, CSS, Javascript, Servlets and JSP*

# Agenda

- HTML Elements
- HTML Forms
- Javascript Introduction
- Javascript Functions
- Form Validation
- CSS Basics
- JSON
- Introduction to Servlets
- Servlets Life cycle

# Agenda

- Handling Request and Response in Servlets
- Forward and Include
- MVC pattern
- Session Management
- Introduction to JSP
- Implicit Objects in JSP
- Custom Tags
- End to End application development with AJAX and REST

# HTML Features

- **Publish** online documents with headings, text, tables, lists, photos, etc.
- **Retrieve** online information via hypertext links, at the click of a button.
- **Design forms** for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- **Include** spread-sheets, video clips, sound clips, and other applications directly in their documents.

# Types of HTML tags

- Layout Tags
  - Basic Tags
  - Structural Tags
- Semantic Tags
  - Presentation Tags
  - Links & Graphics.
  - Lists.
  - Dividers
  - Backgrounds and Colors.
  - Special Characters.
- Application Tags
  - Forms
  - Tables
  - Frames
- Logical Tags
  - Emphasized Tag
  - Strong Tag
  - Code Tag

# HTML Forms

Mode of accepting input from user

## Container: Form

```
<FORM NAME="" ACTION="" METHOD=""></FORM>
```

## Components: Form Elements

- These are the data fields in the form, such as text fields and checkboxes
- The <INPUT> tag can be used to create:
  - Text boxes                      - Radio buttons
  - Check boxes                      - Submit buttons
  - Reset buttons                      - Password
  - Generic buttons
- There are other tags like TEXTAREA and SELECT for other purposes.

# CSS

- CSS stands for Cascading Style Sheet
- Used to style HTML elements
- CSS styles can be applied to
  - all the selector or
  - group of selectors sharing common class name or
  - to an unique selector having an unique id

# Javascript

- JavaScript is an easy to use object scripting language.
- Scripts are run by the Web browser.
- JavaScript can be combined directly with HTML



# Javascript Functions

- Functions are one of the fundamental building blocks in JavaScript.
- A function definition looks as follows:

```
function gcd(m,n) {  
    return  n > 0    ?  gcd(n,m%n)  :  m  ;  
}
```

# Form Validation

- Javascript functions can be called to process the form
- If the forms have invalid inputs then the functions can return false so that inputs aren't submitted
- You will handle different kind of events while validating forms like submit, click, input and so on

# JSON

- JSON stands for JavaScript Object Notation
- It is a lightweight data-interchange format.
- It is easy for humans to read and write.

# JSON

- In JSON, they take on these forms:
  - An *object*  
 $\{\text{"property"} : \text{value}, \dots \}$
  - An *array*  
 $[\{\text{"property"} : \text{value}, \dots\}, \{\dots \}]$

# Servlets

Servlet Life Cycle, Request & Response,  
Form Handling

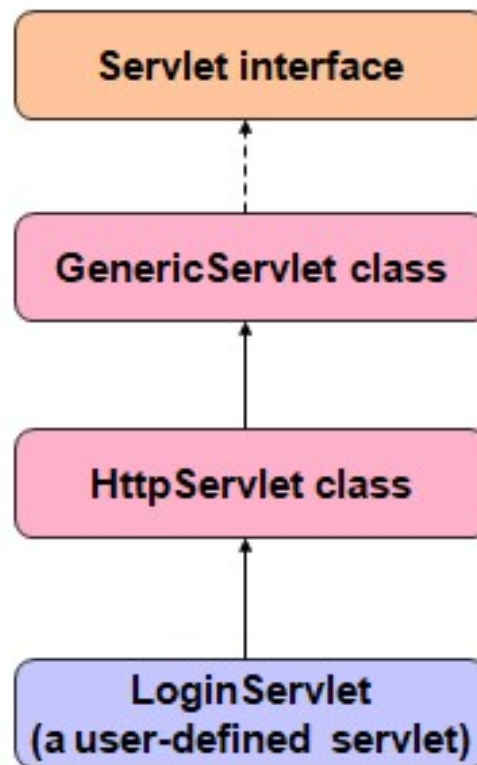
# What are Servlets ?

- 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



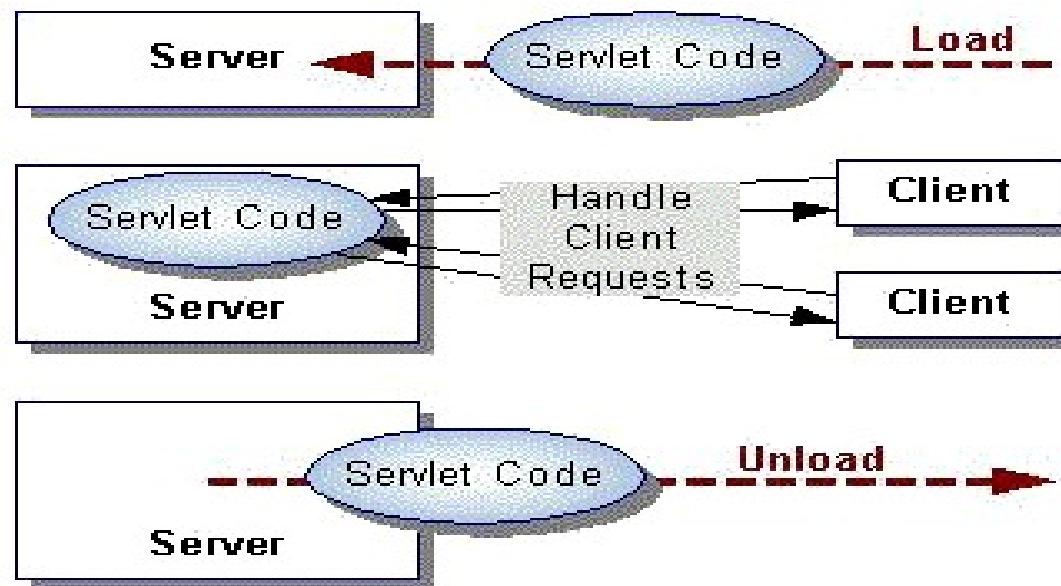


# Servlet Interface

- Provides methods that manage the servlet and its communications with clients
  - `init(ServletConfig)`
  - `service(ServletRequest, ServletResponse)`
  - `destroy()`
  - `getServletConfig()`
  - `getServletInfo()`

# Servlet Life Cycle

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



# Servlet Life Cycle 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()

# HttpServlet

- It is a servlet used specifically when you are communicating with HTTP protocol
- HttpServlet handles the HTTP protocol in the service method
- You can override the methods like doGet or doPost of HttpServlet based on the type of request uses makes

# HttpServletRequest

- HttpServletRequest object incorporates any communication from client to servlet
- Methods
  - `getParameter(String pname)`
  - `getParameterNames()`
  - `getParameterValues(String pname)`

# HttpServletResponse

- HttpServletResponse object incorporates any communication from servlet to client
- Methods
  - getWriter
  - setContentType
  - sendRedirect

# Handling Form Data

- Write a Servlet that retrieves form parameters from the HTML form
- Simpleform.html

```
<html>
<head><title>Simple Form</title></head>
<body>
    <form method="post" action="SimpleFormServlet">
        <h3>Enter user details</h3>
        <br>Name: <input type="text" name="userName" />
        <br>Address: <input type="text" name="userAddress" />
        <input type="submit" value="Submit" />
    </form>
</body>
```


# SimpleFormServlet

- SimpleFormServlet's doPost method retrieves request parameters such as user's name and address having a single value from the form

```
public class SimpleFormServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String name = request.getParameter("userName");
        String address = request.getParameter("userAddress");
        out.println("<html>");
        out.println("<h1>" + "User Details" + "</h1>");
        out.println("<p>The Name you entered was: " + name + "</p>");
        out.println("<p> The Address you entered was: " + address + "</p>");
    }
}
```




# Demo for handling form data

Address  http://localhost:10000/ServletsModularization/Simpleform.html

## Enter user details

Name:   
Address:

Address  http://localhost:10000/ServletsModularization/SimpleFormServlet

## User Details

The Name you entered was: Anny

The Address you entered was: 24th Main Bangalore

# RequestDispatcher

- Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server.

- Below code instantiates RequestDispatcher

*RequestDispatcher rd = request.getRequestDispatcher(url)*

- RequestDispatcher has two methods
  - forward
  - include

# RequestDispatcher methods

- `forward(ServletRequest req, ServletResponse res)`
  - Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
- `include(ServletRequest req, ServletResponse res)`
  - Includes the content of a resource (servlet, JSP page, HTML file) in the response.

# Session Management

- Http is a stateless protocol
- A mechanism is needed to maintain state
- Session tracking is keeping track of what has gone before in a particular conversation

# Session Management Mechanism

- Three different session tracking mechanisms of passing “session id”
  - Cookies – You can use a single cookie containing a session id
  - URL rewriting - You can append a unique ID after the URL to identify user
  - Hidden <form> fields – You can use these to store a unique ID

# Session Management Mechanism (Contd.)

- HttpSession
- Servlet container creates HttpSession object
- Contains Methods to
  - View and manipulate information about a session, such as session identifier, creation time, and last accessed time
  - Bind objects to sessions, allowing user information to persist across multiple user connections

# JSP

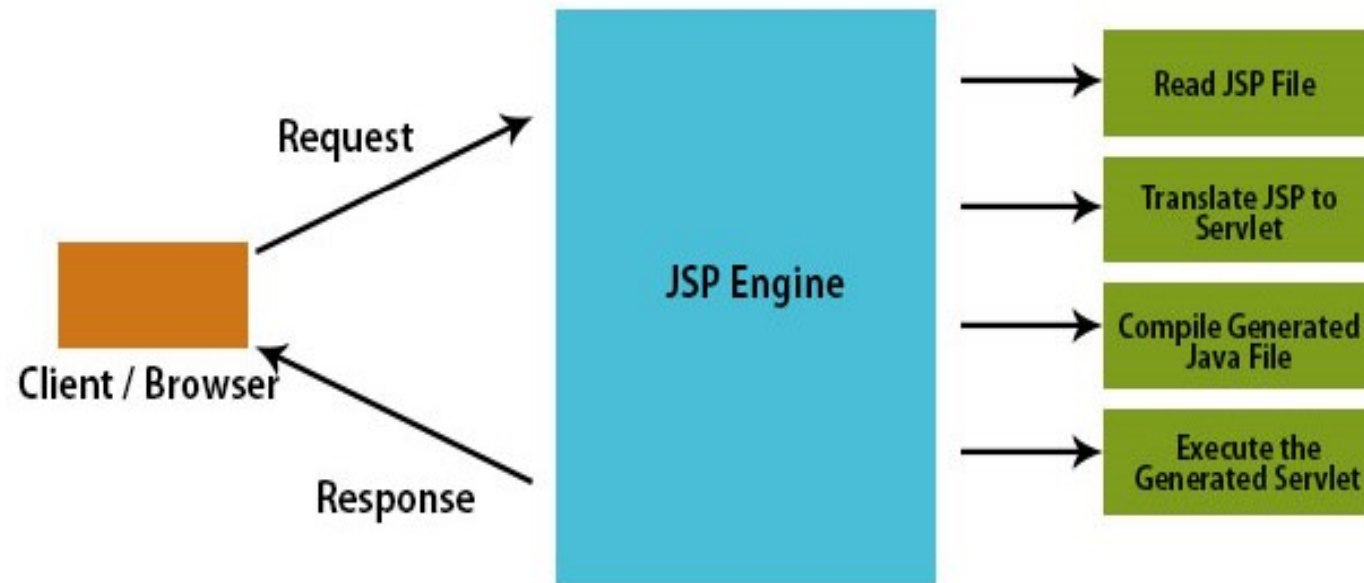
*Simplifies writing HTML at the server side*

# JSP

- JSP stands for Java Server Pages
- Technology invented by Sun to allow the easy creation of server side HTML pages.
- JSP lies in the presentation tier on the web server, with



# JSP Architecture



# JSP Elements

- The four basic elements of a JSP are:
  - Scripting Elements
  - Directive Elements
  - Action Elements
  - Implicit objects

# Scripting Elements

- Lets you insert Java code into the servlet that will be generated from the current JSP page.
- There are three forms:
  - **Expressions**  
Useful shorthand for printing out strings and contents of variables.
  - **Scriptlets**  
Lets you insert any valid Java code into the JSP.
  - **Declarations**  
Useful for declaring page wide variables and methods(Java) or functions.

# Directive Elements

- Affect the overall structure of the servlet class generated from this JSP
- format: `<%@ directive attribute="value"%>`
- There are three main types of directives:
  - page
  - include
  - taglib

# Action Elements

- Control the behavior of the servlet engine
- You can dynamically insert a file, reuse JavaBeans components
- Available actions include:
  - jsp:include
  - jsp:forward
  - jsp:useBean

# Implicit Objects

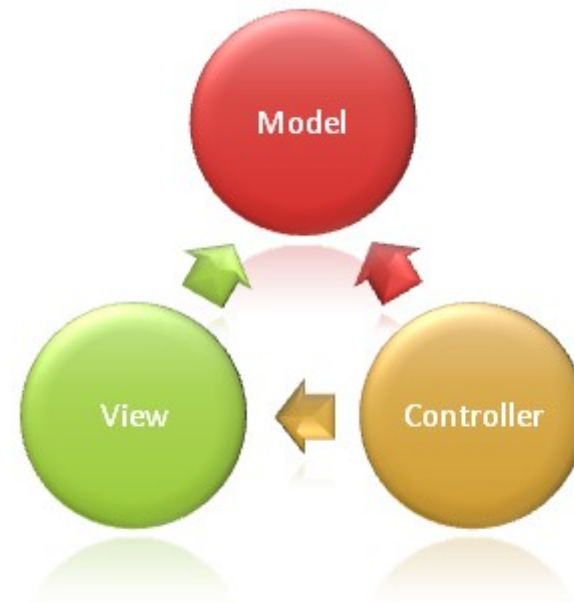
- JSP gives you implicit objects which are available in the JSP page so that developers can use without creating them, some of them are:
  - request
  - out
  - response
  - application
  - session
  - exception

# Custom Tags

- Custom Tags are user defined tags which can do some tasks when its used in the JSP page
- Custom Tags are configured in the .tld files which specifies the *tag names* and *tag handler class*  
`<%@taglib` directive that gives the location of the tld file
- Tag handler class are the classes that will have code which are executed when the tags are parsed

# MVC Design Pattern

- MVC is a design pattern that facilitates development of scalable, flexible applications by separating the application components into three layers:
  - *Model*
  - *View*
  - *Controller*





# Model

- The model is responsible for managing the data of the application.
- It responds to the request from the view and it also responds to instructions from the controller to update itself
- The Model represents the application core (for instance a list of database records).

# View

- The View displays the data (the database records).
- A view requests information from the model, that it needs to generate an output representation.
- MVC is often seen in web applications, where the view is the HTML page.

# Controller

- The Controller is the part of the application that handles user interaction.
- Typically controllers read data from a view, control user input, and send input data to the model.
- It handles the input, typically user actions and may invoke changes on the model and view.

# Advantage of MVC pattern

- MVC pattern helps us achieve loose coupling by dividing the application into the model, view and controller components.
- Since the model and controller are independent of the view, one view technology can be easily swapped for the other, application is not very affected in case of such a swap.