



THIRUTHANGAL NADAR COLLEGE

(Belongs to the Chennaivazh Thiruthangal Hindu Nadar Uravinmurai Dharma Fund)

Selavayal, Chennai-51.

A Self-Financing Co-educational College of Arts & Science

Affiliated to the University of Madras

Accredited with 'B' Grade by NAAC

An ISO 9001: 2015 Certified Institution

NAME OF THE DEPARTMENT: COMPUTER SCIENCE

TOPIC : JAVA SERVER PAGES

STAFF NAME : D. LISSY

Topics

JSP Fundamentals

JSP Scripting Elements

JSP Implicit Objects

JSP Directives

JSP Actions

Servlets & JSPs together

Deploying and Running a JSP Application



Java Server Pages (JSP)

Fundamentals

Java Server Pages are HTML pages embedded with snippets of Java code.

- It is an inverse of a Java Servlet

Four different elements are used in constructing JSPs

- Scripting Elements
- Implicit Objects
- Directives
- Actions

Java Server Pages (JSP)

Architecture

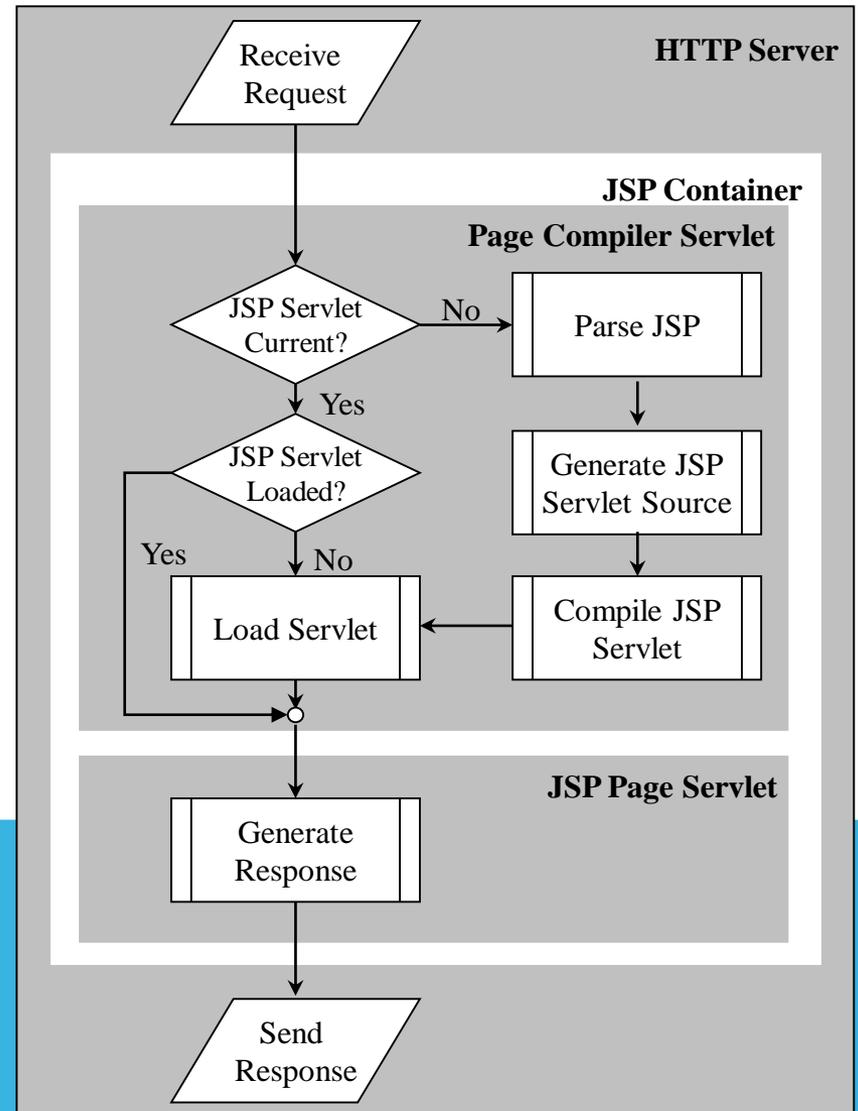
JSPs run in two phases

- Translation Phase
- Execution Phase

In translation phase JSP page is compiled into a servlet

- called JSP Page Implementation class

In execution phase the compiled JSP is processed



Scripting Elements

Types

There are three kinds of scripting elements

- Declarations
- Scriptlets
- Expressions

Declarations

Basics

Declarations are used to define methods & instance variables

- Do not produce any output that is sent to client
- Embedded in `<%!` and `%>` delimiters

Example:

```
<%!  
    Public void jspDestroy() {  
        System.out.println("JSP Destroyed");  
    }  
    Public void jspInit() {  
        System.out.println("JSP Loaded");  
    }  
    int myVar = 123;  
%>
```

The functions and variables defined are available to the JSP Page as well as to the servlet in which it is compiled

Scriptlets

Basics

Used to embed java code in JSP pages.

- Contents of JSP go into `_jspPageservice()` method
- Code should comply with syntactical and semantic construct of java
- Embedded in `<%` and `%>` delimiters

Example:

```
<%  
    int x = 5;  
    int y = 7;  
    int z = x + y;  
%>
```

Expressions

Basics

Used to write dynamic content back to the browser.

- If the output of expression is Java primitive the value is printed back to the browser
- If the output is an object then the result of calling `toString` on the object is output to the browser
- Embedded in `<%=` and `%>` delimiters

Example:

- `<%=“Fred”+ “ “ + “Flintstone” %>`
prints “Fred Flintstone” to the browser
- `<%=Math.sqrt(100)%>`
prints 10 to the browser

Java Implicit Objects

Scope

Implicit objects provide access to server side objects

- e.g. request, response, session etc.

There are four scopes of the objects

- Page: Objects can only be accessed in the page where they are referenced
- Request: Objects can be accessed within all pages that serve the current request.
(Including the pages that are forwarded to and included in the original jsp page)
- Session: Objects can be accessed within the JSP pages for which the objects are defined
- Application: Objects can be accessed by all JSP pages in a given context

Java Implicit Objects

List

request: Reference to the current request

response: Response to the request

session: session associated with current request

application: Servlet context to which a page belongs

pageContext: Object to access request, response, session and application associated with a page

config: Servlet configuration for the page

out: Object that writes to the response output stream

page: instance of the page implementation class

(this)

exception: Available with JSP pages which are error pages

Java Implicit Objects

Example

```
<html>
<head>
  <title>Implicit Objects</title>
</head>
<body style="font-family:verdana;font-size:10pt">
  <p>
    Using Request parameters...<br>
    <b>Name:</b> <%= request.getParameter("name")
      %>
  </p>
  <p>
    <%= out.println("This is printed using the out implicit
      variable"); %>
  </p>
  <p>
    Storing a string to the session...<br>
    <%= application.setAttribute("name", "Meeraj"); %>
    Retrieving the string from session...<br>
    <b>Name:</b> <%= session.getAttribute("name") %>
  </p>
```

```
<p>
  Storing a string to the application...<br>
  <%= application.setAttribute("name", "Meeraj"); %>
  Retrieving the string from application...<br>
  <b>Name:</b>
  <%= application.getAttribute("name") %>
</p>
<p>
  Storing a string to the page context...<br>
  <%= pageContext.setAttribute("name", "Meeraj"); %>
  Retrieving the string from page context...<br>
  <b>Name:</b>
  <%= pageContext.getAttribute("name") %>
</p>
</body>
</html>
```

Example Implicit Objects

Deploy & Run

Save file:

- `$TOMCAT_HOME/webapps/jsp/Implicit.jsp`

Access file

- `http://localhost:8080/jsp/Implicit.jsp?name=Sanjay`

Results of the execution

Using Request parameters...

Name: sanjay

This is printed using the out implicit variable

Storing a string to the session...

Retrieving the string from session...

Name: Meeraj

Storing a string to the application...

Retrieving the string from application...

Name: Meeraj

Storing a string to the page context...

Retrieving the string from page context...

Name: Meeraj

Directives

Basics & Types

Messages sent to the JSP container

- Aids the container in page translation

Used for

- Importing tag libraries
- Import required classes
- Set output buffering options
- Include content from external files

The jsp specification defines three directives

- Page: provide information about page, such as scripting language that is used, content type, or buffer size
- Include – used to include the content of external files
- Taglib – used to import custom actions defined in tag libraries

Page Directives

Basics & Types

Page directive sets page properties used during translation

- JSP Page can have any number of directives
- Import directive can only occur once
- Embedded in `<%@` and `%>` delimiters

Different directives are

- Language: (Default Java) Defines server side scripting language (e.g. java)
- Extends: Declares the class which the servlet compiled from JSP needs to extend
- Import: Declares the packages and classes that need to be imported for using in the java code (comma separated list)
- Session: (Default true) Boolean which says if the session implicit variable is allowed or not
- Buffer: defines buffer size of the jsp in kilobytes (if set to none no buffering is done)

Page Directives

Types con't.

Different directives are (cont'd.)

- **autoFlush:** When true the buffer is flushed when max buffer size is reached (if set to false an exception is thrown when buffer exceeds the limit)
- **isThreadSafe:** (default true) If false the compiled servlet implements SingleThreadModel interface
- **Info:** String returned by the `getServletInfo()` of the compiled servlet
- **errorPage:** Defines the relative URI of web resource to which the response should be forwarded in case of an exception
- **contentType:** (Default text/html) Defines MIME type for the output response
- **isErrorPage:** True for JSP pages that are defined as error pages
- **pageEncoding:** Defines the character encoding for the jsp page

Page Directives

Example

`<%@`

`page language="java"`

`buffer="10kb"`

`autoflush="true"`

`errorPage="/error.jsp"`

`import="java.util.*, javax.sql.RowSet"`

`%>`



Include Directive

Basics

Used to insert template text and JSP code during the translation phase.

- The content of the included file specified by the directive is included in the including JSP page

Example

- `<%@ include file="included.jsp" %>`

JSP Actions

Basics & Types

Processed during the request processing phase.

- As opposed to JSP directives which are processed during translation

Standard actions should be supported by J2EE compliant web servers

Custom actions can be created using tag libraries

The different actions are

- Include action
- Forward action
- Param action
- useBean action
- getProperty action
- setProperty action
- plugIn action

JSP Actions

Include

Include action used for including resources in a JSP page

- Include directive includes resources in a JSP page at translation time
- Include action includes response of a resource into the response of the JSP page
- Same as including resources using `RequestDispatcher` interface
- Changes in the included resource reflected while accessing the page.
- Normally used for including dynamic resources

Example

- `<jsp:include page="includedPage.jsp">`

Includes the the output of `includedPage.jsp` into the page where this is included.

JSP Actions

Forward

Forwards the response to other web specification resources

- Same as forwarding to resources using RequestDispatcher interface

Forwarded only when content is not committed to other web application resources

- Otherwise an IllegalStateException is thrown
- Can be avoided by setting a high buffer size for the forwarding jsp page

Example

- `<jsp:forward page="Forwarded.html">`

Forwards the request to Forwarded.html

JSP Actions

Param

Used in conjunction with Include & Forward actions to include additional request parameters to the included or forwarded resource

Example

```
<jsp:forward page="Param2.jsp">
```

```
    <jsp:param name="FirstName" value="Sanjay">
```

```
</jsp:forward>
```

- This will result in the forwarded resource having an additional parameter FirstName with a value of Sanjay

JSP Actions

useBean

Creates or finds a Java object with the defined scope.

- Object is also available in the current JSP as a scripting variable

Syntax:

```
<jsp:useBean id="name"  
scope="page | request | session | application"  
class="className" type="typeName" |  
bean="beanName" type="typeName" |  
type="typeName" />
```

- At least one of the type and class attributes must be present
- We can't specify values for both the class and bean name.

Example

```
<jsp:useBean id="myName" scope="request" class="java.lang.String">  
    <% firstName="Sanjay"; %>  
</jsp:useBean>
```

JSP Actions

get/setProperty

getProperty is used in conjunction with **useBean** to get property values of the bean defined by the **useBean** action

Example (getProperty)

- `<jsp:getProperty name="myBean" property="firstName" />`
- Name corresponds to the id value in the **useBean**
- Property refers to the name of the bean property

setProperty is used to set bean properties

Example (setProperty)

- `<jsp:setProperty name="myBean" property="firstName" value="Sanjay"/>`
- Sets the name property of **myBean** to **Sanjay**
`Example (setProperty)`
- `<jsp:setProperty name="myBean" property="firstName" param="fname"/>`
- Sets the name property of **myBean** to the request parameter **fname**
- `<jsp:setProperty name="myBean" property="*">`
- Sets property to the corresponding value in request

JSP Actions

plugin

Enables the JSP container to render appropriate HTML (based on the browser type) to:

- Initiate the download of the Java plugin
- Execution of the specified applet or bean

plugin standard action allows the applet to be embedded in a browser neutral fashion

Example

```
<jsp: plugin type="applet" code="MyApplet.class" codebase="/">
  <jsp:params>
    <jsp:param name="myParam" value="122"/>
  </jsp:params>
  <jsp:fallback><b>Unable to load applet</b></jsp:fallback>
</jsp:plugin>
```