**<jsp:forward> Tag in JSP**

The <jsp:forward> tag is used to forward a client's request from one JSP page (or Servlet) to another resource (which can be another JSP page, Servlet, or HTML file) on the server. Once the request is forwarded, control is transferred to the new resource, and the current page stops processing. This is helpful when you want to centralize some logic or want a clean separation of concerns in your application.

**Key Points:**

* **Once the forward happens**, the current JSP page stops executing any further code.
* **Request parameters** can also be forwarded by setting them as attributes before forwarding or using <jsp:param>.
* **Forward happens on the server**, so the client (browser) is unaware of the forward operation. The browser URL remains the same.

**Syntax of <jsp:forward>:**

jsp

Copy code

<jsp:forward page="relativeURL" />

**Simple Example Program**

**Scenario:**

You have a login page, and based on user input, you forward the request either to a success page or back to the login page if the credentials are wrong.

**File 1: login.jsp**

jsp

Copy code

<html>

<head>

<title>Login Page</title>

</head>

<body>

<h2>Login Form</h2>

<form action="validate.jsp" method="post">

Username: <input type="text" name="username" /><br />

Password: <input type="password" name="password" /><br />

<input type="submit" value="Login" />

</form>

</body>

</html>

**File 2: validate.jsp**

jsp

Copy code

<%

// Retrieve form data

String username = request.getParameter("username");

String password = request.getParameter("password");

// Simple validation (in real applications, validation would be more robust)

if ("admin".equals(username) && "password123".equals(password)) {

// If the username and password are correct, forward to success page

request.setAttribute("user", username);

%>

<jsp:forward page="welcome.jsp" />

<%

} else {

// If credentials are wrong, forward back to the login page with an error message

request.setAttribute("errorMessage", "Invalid username or password.");

%>

<jsp:forward page="login.jsp" />

<%

}

%>

**File 3: welcome.jsp**

jsp

Copy code

<html>

<head>

<title>Welcome</title>

</head>

<body>

<h2>Welcome Page</h2>

<p>Hello, <%= request.getAttribute("user") %>! You have successfully logged in.</p>

</body>

</html>

**Explanation:**

1. **login.jsp:**
   * This page displays a simple login form with fields for username and password.
   * When the form is submitted, the data is sent to validate.jsp.
2. **validate.jsp:**
   * It checks the username and password values.
   * If the user enters admin as the username and password123 as the password, the request is forwarded to welcome.jsp.
   * If the credentials are invalid, the request is forwarded back to login.jsp with an error message using the <jsp:forward> tag.
   * Notice how no further code runs once the <jsp:forward> action happens.
3. **welcome.jsp:**
   * Displays a welcome message that includes the username (admin in this case) forwarded from validate.jsp.

By using the <jsp:forward> tag, you effectively control the navigation between pages based on logic or user input. The browser's URL remains unchanged since the forward happens entirely on the server.

**<jsp:include> Tag in JSP**

The <jsp:include> tag is used to include the content of another resource (such as a JSP file, HTML file, or servlet) into the current JSP page. The included page can be static or dynamic. The content of the included page is executed and added to the main JSP page's output.

**Key Points:**

* **Dynamic inclusion**: The included content is evaluated at request time (i.e., when the main JSP is requested). If the included resource changes, the next request will reflect those changes.
* **Modularity**: It helps in breaking down the application into reusable components like headers, footers, and sidebars.
* **Retaining request and response**: The request and response objects are passed to the included page, allowing data sharing between the pages.

**Syntax of <jsp:include>:**

jsp

Copy code

<jsp:include page="relativeURL" flush="true" />

* **page**: Specifies the relative URL of the resource to be included.
* **flush**: If set to true, it ensures that the output buffer of the calling page is flushed before including the content. This attribute is optional and often defaults to true.

**Simple Example Program**

**Scenario:**

You want to create a webpage with a common header and footer that can be reused across multiple pages.

**File 1: header.jsp (Common Header)**

jsp

Copy code

<html>

<head>

<title>My Website</title>

</head>

<body>

<header>

<h1>Welcome to My Website</h1>

<nav>

<a href="home.jsp">Home</a> |

<a href="about.jsp">About Us</a> |

<a href="contact.jsp">Contact Us</a>

</nav>

<hr>

</header>

**File 2: footer.jsp (Common Footer)**

jsp

Copy code

<footer>

<hr>

<p>&copy; 2024 My Website. All Rights Reserved.</p>

</footer>

</body>

</html>

**File 3: home.jsp (Main Page that Includes Header and Footer)**

jsp

Copy code

<jsp:include page="header.jsp" />

<h2>Home Page</h2>

<p>Welcome to the homepage of my website. Here, you'll find various articles, news, and updates.</p>

<jsp:include page="footer.jsp" />

**File 4: about.jsp (Another Page that Includes Header and Footer)**

jsp

Copy code

<jsp:include page="header.jsp" />

<h2>About Us</h2>

<p>We are a company dedicated to providing the best services to our clients. Our mission is to deliver excellence.</p>

<jsp:include page="footer.jsp" />

**Explanation:**

1. **header.jsp:**
   * This file contains the common header for the website. It includes the page title, a welcome message, and a navigation menu with links to other pages.
   * The header is structured with an HTML <header> element and a horizontal line (<hr>).
2. **footer.jsp:**
   * This file contains the common footer for the website. It adds a copyright message and a horizontal line (<hr>).
   * The footer is structured using an HTML <footer> element and closes the <body> and <html> tags.
3. **home.jsp:**
   * The main content of the home page is displayed between the included header.jsp and footer.jsp files.
   * The <jsp:include> tag is used to include the header and footer dynamically.
4. **about.jsp:**
   * Another page on the website that also uses the same header.jsp and footer.jsp files to create a consistent layout across pages.

**How It Works:**

* The **header.jsp** and **footer.jsp** are included in both the home.jsp and about.jsp files using the <jsp:include> tag.
* This allows you to reuse the same header and footer content across multiple pages without repeating the code.
* If you make a change to the header or footer, it will automatically be reflected across all pages that include them.

By using <jsp:include>, you can create modular and maintainable JSP pages where common elements like headers, footers, or sidebars are managed centrally, making updates easier and ensuring consistency across your web application.

**<jsp:useBean>** <jsp:getProperty> <jsp:setProperty> <jsp:param>**Tag in JSP**

The <jsp:useBean> tag in JSP is used to create or access a **JavaBean** in a JSP page. A **JavaBean** is a reusable Java class that follows specific conventions: it has a no-argument constructor, getter and setter methods for its properties, and is serializable. JavaBeans are commonly used to encapsulate data and logic and can be easily shared across different parts of the application.

**Key Points:**

* **JavaBean Creation/Reuse**: The <jsp:useBean> tag either locates an existing JavaBean (if it's already been created and placed in the scope) or creates a new instance if it doesn't exist.
* **Scope**: The bean can be stored in one of four scopes:
  + **page**: The bean is available only on the current JSP page.
  + **request**: The bean is available for the entire request, including forwarding to another JSP or servlet.
  + **session**: The bean is stored in the session and is available across multiple requests.
  + **application**: The bean is stored in the application context and is available across the entire web application.
* **Properties**: You can use <jsp:setProperty> to set bean properties and <jsp:getProperty> to retrieve them.

**Syntax of <jsp:useBean>:**

jsp

Copy code

<jsp:useBean id="beanInstance" class="fullyQualifiedClassName" scope="scope" />

* **id**: The identifier of the bean instance in the JSP page.
* **class**: The fully qualified name of the JavaBean class.
* **scope**: Specifies the scope (page, request, session, or application). If not provided, the default is page.

**Simple Example Program**

**Scenario:**

You want to create a simple JSP page where a user submits their name and age, and the page displays the information using a JavaBean.

**JavaBean Class: Person.java**

package com.example;

public class Person {

private String name;

private int age;

// No-argument constructor

public Person() {}

// Getter and Setter for name

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// Getter and Setter for age

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

**File 1: input.jsp (User Input Page)**

jsp

Copy code

<html>

<head>

<title>Person Information</title>

</head>

<body>

<h2>Enter Your Information</h2>

<%-- Anything you want to be commented --%>

<form action="display.jsp" method="post">

Name: <input type="text" name="name" /><br /><br />

Age: <input type="text" name="age" /><br /><br />

<input type="submit" value="Submit" />

</form>

</body>

</html>

**File 2: display.jsp (Displaying Bean Properties)**

jsp

Copy code

<jsp:useBean id="person" class="com.example.Person" scope="request" />

<jsp:setProperty name="person" property="name" param="name" />

<jsp:setProperty name="person" property="age" param="age" />

<html>

<head>

<title>Display Information</title>

</head>

<body>

<h2>Person Information</h2>

<p>Name: <jsp:getProperty name="person" property="name" /></p>

<p>Age: <jsp:getProperty name="person" property="age" /></p>

</body>

</html>

**Explanation:**

1. **Person.java:**
   * This is a simple JavaBean class with two properties: name and age. It has getter and setter methods for each property, and a no-argument constructor, as required by the JavaBean specification.
2. **input.jsp:**
   * This JSP page contains a form where the user can input their name and age. When the form is submitted, it sends the data to display.jsp.
3. **display.jsp:**
   * **<jsp:useBean>**: Creates or reuses a bean of the class com.example.Person. If the bean is not already present in the request scope, it is created.
   * **<jsp:setProperty>**: The name and age properties of the person bean are set using the form parameters (param="name" and param="age" refer to the form fields).
   * **<jsp:getProperty>**: Retrieves the values of the name and age properties from the bean and displays them on the page.

**How It Works:**

* When the user fills out the form in input.jsp and submits it, the data is sent via POST to display.jsp.
* In display.jsp, the JavaBean (person) is either created or reused, and its name and age properties are set based on the form data.
* The values of the bean properties are then retrieved and displayed to the user using <jsp:getProperty>.

This is a simple example demonstrating how to use JavaBeans in JSP pages with the <jsp:useBean>, <jsp:setProperty>, and <jsp:getProperty> tags. It helps to separate the data (in JavaBeans) from the presentation logic (in JSP).

Bottom of Form