How to create a Bean class?

1. Declared using public
2. Must implements Serializable
3. Zero argument constructor.
4. Data members must be declared as private
5. Setter and getter methods for all datamembers.

Action Tags

Introduced in JSP 1.1

|  |  |
| --- | --- |
| <jsp:usebean> | Enables the java bean component |
| <jsp:getProperty> | Get property values from Java bean component and adds to the response |
| <jsp:setProperty> | Sets the java bean property values |
| <jsp:include> | Includes the response from the JSP page while processing |
| <jsp:forward> | Forwards the processing of request to a jsp page |
| <jsp:param> | Adds the parameter value to a request handed off using <jsp:include> or <jsp:forward> |
| <jsp:attributes> | Sets the value of the action element |

<jsp:useBean>

* Used to create a bean object.
* Stores the created bean in a scope.
* Also used to get the existing bean object from scope.
* Attributes are:
  + Id: reference of the bean class
  + class: contains the bean class name
  + scope: contains the scope

Example: 1

<jsp:useBean class=”Student” id = “stud” scope=”session”>

Verifies whether student class with object id is present in session scope or not.

If it is present then it gives the reference of that object to a reference variable stud.

If not present in session scope with id stud, then creates and adds to session scope.

Equavalent Java code

Student stud;

Stud = (Student)session.getAttribute(“stud”);

If(stud==null)

{

stud = new Student();

session.setAttribute(“stud”,stud);

}

Example: 2(Super class reference – sub class object)

<jsp:useBean class=”Student” type = “person” id = “stud” scope=”session”>

Equavalent Java code

Person stud = (Person)session.getAttribute(“stud”);

If(stud==null)

{

stud = new Student();

session.setAttribute(“stud”,stud);

}

Example: 3(Using Bean class Name)

<jsp:useBean beanName=”Student” id = “stud” scope=”session”>

Equavalent Java code

Student stud;

Stud = (Student)session.getAttribute(“stud”);

If(stud==null)

{

stud = new Student();

session.setAttribute(“stud”,stud);

}

<jsp:setProperty>

* Used to set datamembers of bean.
* Attributes are:
  + name: name of the bean object reference variable which is declared in <jsp:useBean> tag using id attribute.
  + property: name of the bean class data member which has to be set.
  + value: contains the value which has to set to the java bean class data member. This is optional.
  + param: contains the name of the request parameter whose value has to set to the bean data member.

|  |
| --- |
| Example: 1  <jsp:useBean class = “Student” id = “stud” />  <jsp:setProperty name=”stud” property=”htno” value=”101”/>  Equavalent Java Code  Student stud = (Student)session.getAttribute(“stud”);  If(stud==null)  {  Stud = new Student();  session.setAttribute(“stud”,stud);  }  stud.setHtno(101); |
| Example: 2  <jsp:useBean class = “Student” id = “stud” />  <jsp:setProperty name=”stud” property=”htno” param=”htno”/>  Equavalent Java Code  Student stud = (Student)session.getAttribute(“stud”);  If(stud==null)  {  Stud = new Student();  session.setAttribute(“stud”,stud);  }  String temp = request.getParameter(“htno”);  Int htno = Integer.parseInt(temp);  stud.setHtno(htno); |

Note:

1. Whenever request parameter name and property name are same, we need not to use param attribute.

ie. Example 2 can be written as

<jsp:useBean class = “Student” id = “stud” />

<jsp:setProperty name=”stud” property=”htno” />

1. If all the request parameter names are same as data members then use property=”\*”

<jsp:useBean class = “Student” id = “stud” />

<jsp:setProperty name=”stud” property=”\*” />

<jsp:getProperty>

Used to get the data members of the bean object.

Attributes are:

name : name of the bean class reference variable

property : name of the bean object data member

Example:

<jsp:useBean class = “Student” id = “stud” scope = “session”/>

<jsp:getProperty name=”stud” property=”htno” />

Equivalent java code

Student stud = (Student)session.getAttribute(“stud”);

out.println(stud.getHtno());

<jsp:include>

Used to include the content dynamically

Two attributes:

page --- url of the target jsp page

flush --- optional, takes a boolean value which is true or false. Default is false

Note: <%@ page include=”file” %> ----- is called static include.

Example:

jspPage1.jsp

<h1> Hai....This is page1 </h1>

<jsp:include page=”jspPage2.jsp”>

<h1> Bye.......Page1 </h1>

jspPage2.jsp

<h1> Hai....This is page2 </h1>

<h1> Bye.......Page1 </h1>

Equavalent Java code

out.println(“<h1> Hai.....This is Page1 </h1>”);

RequestDispatcher rd = request.getRequestDispatcher(“jspPage2.jsp”);

rd.include(request,response);

out.println(“<h1> Bye....This is Page1 </h1>”);

<jsp:forward>

Used to forward request from one jsp to another jsp.

Only one attribute

page --- url of the target jsp

<jsp:forward page=”view.jsp”>

Equavalent Java code

RequestDispatcher rd = request.getRequestDispatcher(“view.jsp”);

rd.forward(request,response);

<jsp:param>

Used to send request parameters while forwarding the request to other jsp.

Must be used inside <jsp:include> or <jsp:forward> tag.

Two parameters that are mandatory:

name --- name of the parameter

value --- value of the parameter

Example:

Page1.jsp

<jsp:include page = “/page2.jsp”>

<jsp:param name = “param1” value=”value1”>

<jsp:param name = “param2” value=”value2”>

</jsp:include>

Page2.jsp

Param1 = <%= request.getParameter(“param1”) %>

Param2 = <%= request.getParameter(“param2”) %>

RequestDispatcher rs = request.getRequestDispatcher(“/page2.jsp?param1=value1&param2=value”);

**Expression Language**

Expression Language(EL) was added to JSP 2.0 specification.

The purpose of EL is to produce scriptless JSP pages.

The syntax of EL in a JSP is as follows: ${expr}

Here **expr** is a valid EL expression.

**EL Implicits Objects**

The following are the implicit object in EL

|  |  |  |
| --- | --- | --- |
|  | **Implicit Object** | **Description** |
| **1** | **pageContext** | Represent the PageContext object.  pageContext.setAttribute(...,....,.....); |
| **2** | **pageScope** | It is used to access the value of variable which is set in the **Page** scope  page.setAttribute(...,....,.....); |
| **3** | **requestScope** | It is used to access the value of variable which is set in the **request** scope.  request.setAttribute(...,....,.....); |
| **4** | **sessionScope** | It is used to access the value of variable which is set in the **session** scope  session.setAttribute(...,....,.....); |
| **5** | **applicationScope** | It is used to access the value of variable which is set in the **application** scope  application.setAttribute(...,....,.....); |
| **6** | **Param** | Map a request parameter name to a single value  ${param.parameter} |
| **7** | **paramValues** | Map a request parameter name to corresponding string arrays.  ${paramValue.parameter[index]}  If blank is printed means it is array index out of bounds exception |
| **8** | **Header** | Map containing header names and single string values. |
| **9** | **headerValues** | Map containing header names to corresponding string arrays. |
| **10** | **Cookie** | Map containing cookie names and single string values. |
| **11** | **Init param** | Used to read context initialization parameters  Instead of application.getInitParameter(“.....”) |

Example 1:

**index.jsp**

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

Name <input type="text" name="user" >

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

</form>

**welcome.jsp**

<html>

<head>

<title>Welcome Page</title>

</head>

<body>

<h1>Welcome ${param.name}</h1>

</body>

</html>

Example 2(Application Scope)

index.jsp

<html>

<head>

</head>

<body>

<%

application.setAttribute("author", "Chaitanya");

application.setAttribute("Site", "BeginnesBook.com");

%>

<a href="display.jsp">Click</a>

</body>

</html>

display.jsp

<html>

<head>

<title>Display Page</title>

</head>

<body>

${applicationScope.author}<br>

${applicationScope.Site}

</body>

</html>

|  |  |
| --- | --- |
| **Arithmetic Operation** | **Operator** |
| Addition | + |
| Subtraction | - |
| Multiplication | \* |
| Division | / and div |
| Remainder | % and mod |

**Logical and Relational Operator available in EL**

The following are the logical operator and Comparators avilable in EL.

|  |  |
| --- | --- |
| **Logical and Relational Operator** | **Operator** |
| Equals | == and eq |
| Not equals | != and ne |
| Less Than | < and lt |
| Greater Than | > and gt |
| Greater Than or Equal | >= and ge |
| Less Than or Equal | <= and le |
| And | && and and |
| Or | || and or |
| Not | ! and not |