JSP

Implicit Objects

Table of Contents

There are **9 jsp implicit objects**. These objects are *created by the web container* that are available to all the jsp pages. The available implicit objects are out, request, config, session, application etc.
A list of the 9 implicit objects is given below:

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable

JSP out implicit object

For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter. In case of servlet you need to write:

1. PrintWriter out=response.getWriter();
   But in JSP, you don't need to write this code.

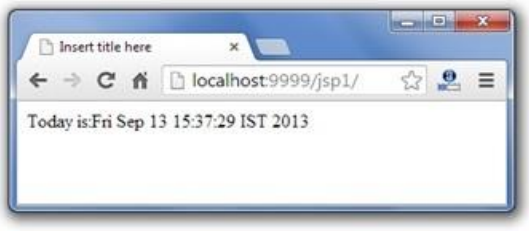
Example of out implicit object

In this example we are simply displaying date and time.

index.jsp

```
1. <html>
2. <body>
3. <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
4. </body>
5. </html>
```

Output



javatpoint.com

JSP request implicit object

The **JSP request** is an implicit object of type HttpServletRequest i.e. created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.

It can also be used to set, get and remove attributes from the jsp request scope.

Let's see the simple example of request implicit object where we are printing the name of the user with welcome message.

Example of JSP request implicit object

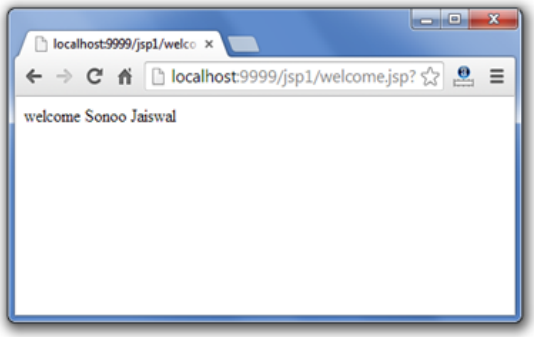
index.html

```
1. <form action="welcome.jsp">
2. <input type="text" name="uname">
3. <input type="submit" value="go"><br/>
4. </form>
```

welcome.jsp

```
1. <%
2. String name=request.getParameter("uname");
3. out.print("welcome "+name);
4. %>
```

Output



JSP response implicit object

In JSP, response is an implicit object of type HttpServletResponse. The instance of HttpServletResponse is created by the web container for each jsp request.
It can be used to add or manipulate response such as redirect response to another resource, send error etc.

Let's see the example of response implicit object where we are redirecting the response to the Google.

Example of response implicit object

index.html

```
1. <form action="welcome.jsp">
2. <input type="text" name="uname">
3. <input type="submit" value="go"><br/>
4. </form>
```

welcome.jsp

```
1. <%
2. response.sendRedirect("http://www.google.com");
3. %>
```

Output

JSP config implicit object

In JSP, config is an implicit object of type *ServletConfig*.
This object can be used to get initialization parameter for a particular JSP page. The config object is created by the web container for each jsp page.

Generally, it is used to get initialization parameter from the web.xml file.

Example of config implicit object:

index.html

```
1. <form action="welcome">
2. <input type="text" name="uname">
3. <input type="submit" value="go"><br/>
4. </form>
```

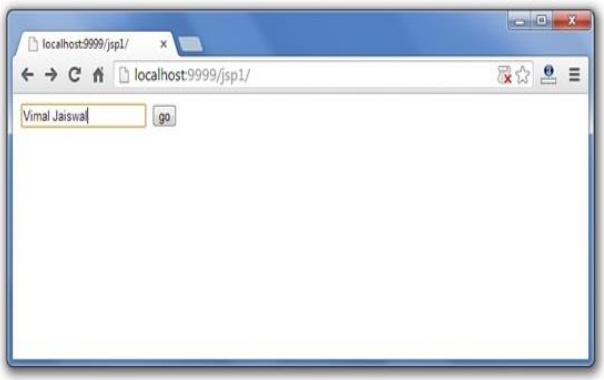
web.xml file

```
1. <web-app>
2.   <servlet>
3. <servlet-name>sonoojaiswal</servlet-name>
4. <jsp-file>/welcome.jsp</jsp-file>
5. <init-param>
6. <param-name>dname</param-name>
7. <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
8. </init-param>
9. </servlet>
10.
11. <servlet-mapping>
12. <servlet-name>sonoojaiswal</servlet-name>
13. <url-pattern>/welcome</url-pattern>
14. </servlet-mapping>
15.
16. </web-app>
```

welcome.jsp

```
<%
out.print("Welcome "+request.getParameter("uname"));

String driver=config.getInitParameter("dname");
out.print("driver name is="+driver);
%>
```

Output



JSP application implicit object

In JSP, application is an implicit object of type *ServletContext*.
The instance of ServletContext is created only once by the web container when application or project is deployed on the server.
This object can be used to get initialization parameter from configuaration file (web.xml).
It can also be used to get, set or remove attribute from the application scope.

This initialization parameter can be used by all jsp pages.

Example of application implicit object:

index.html

```
<form action="welcome">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

web.xml file

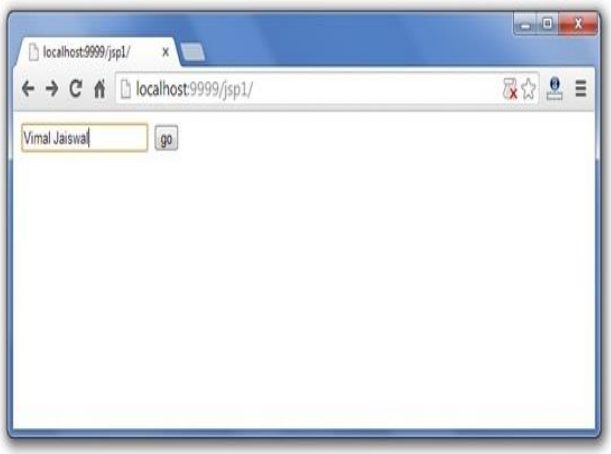
```
<web-app>

<servlet>
<servlet-name>sonoojaiswal</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>
</servlet>

<servlet-mapping>
<servlet-name>sonoojaiswal</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>
```

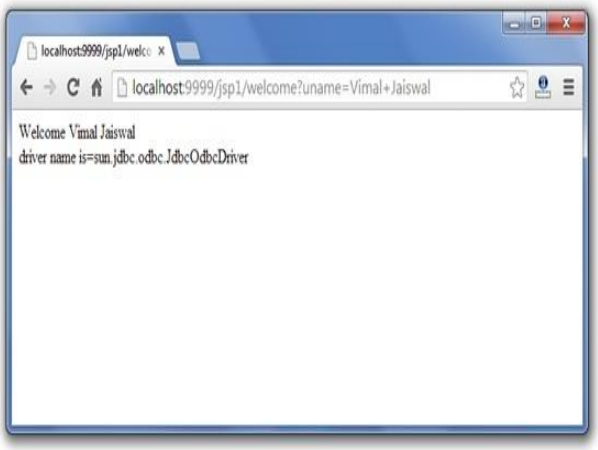
```
12.
13. <context-param>
14. <param-name>dname</param-name>
15. <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
16. </context-param>
17.
18. </web-app>
```

welcome.jsp

```
1. <%
2.
3. out.print("Welcome "+request.getParameter("uname"));
4.
5. String driver=application.getInitParameter("dname");
6. out.print("driver name is="+driver);
7.
8. %>
```

Output





JSP session implicit object

In JSP, session is an implicit object of type HttpSession. The Java developer can use this object to set,get or remove attribute or to get session information.

Example of session implicit object

index.html

```
1. <html>
2. <body>
3. <form action="welcome.jsp">
4. <input type="text" name="uname">
5. <input type="submit" value="go"><br/>
6. </form>
7. </body>
8. </html>
```

welcome.jsp

```
1. <html>
2. <body>
3. <%
4.
5. String name=request.getParameter("uname");
6. out.print("Welcome "+name);
7.
8. session.setAttribute("user",name);
9.
10. <a href="second.jsp">second jsp page</a>
11.
12. %>
13. </body>
14. </html>
```

second.jsp

```
1. <html>
2. <body>
3. <%
4.
5. String name=(String)session.getAttribute("user");
6. out.print("Hello "+name);
7.
8. %>
9. </body>
10. </html>
```

Output

JSP pageContext implicit object

It is an instance of **javax.servlet.jsp.PageContext**. Using this object you can find attribute, get attribute, set attribute and remove attribute at any of the below levels –

1. JSP Page – Scope: PAGE_CONTEXT
2. HTTP Request – Scope: REQUEST_CONTEXT
3. HTTP Session – Scope: SESSION_CONTEXT
4. Application Level – Scope: APPLICATION_CONTEXT

Methods of pageContext Implicit Object

1. **Object findAttribute (String AttributeName):** This method searches for the specified attribute in all four levels in the following order – Page, Request, Session and Application. It returns NULL when no attribute found at any of the level.

2. **Object getAttribute (String AttributeName, int Scope):** It looks for an attribute in the specified scope. This method is similar to findAttribute method; the only difference is that findAttribute looks in all the four levels in a sequential order while getAttribute looks in a specified scope.
For e.g. – In the below statement the getAttribute method would search for the attribute "BeginnersBook" in **Session scope (or Session level/layer).** If it finds the attribute it would assign it to Object obj else it would return Null.
```
Object obj = pageContext.getAttribute("BeginnersBook", PageContext.SESSION_CONTEXT);
```
Similarly the method can be used for other scopes too –
```
Object obj = pageContext.getAttribute("BeginnersBook", PageContext. REQUEST_CONTEXT);
Object obj = pageContext.getAttribute("BeginnersBook", PageContext. PAGE_CONTEXT);
Object obj = pageContext.getAttribute("BeginnersBook", PageContext. APPLICATION_CONTEXT);
```

3. **void removeAttribute(String AttributeName, int Scope):** This method is used to remove an attribute from a given scope. For example – The below JSP statement would remove an Attribute "MyAttr" from page scope.
```
pageContext.removeAttribute("MyAttr", PageContext. PAGE_CONTEXT);
```

4. **void setAttribute(String AttributeName, Object AttributeValue, int Scope):** It writes an attribute in a given scope. Example – Below statement would store an Attribute "mydata" in application scope with the value "This is my data".
```
pageContext.setAttribute("mydata", "This is my data", PageContext. APPLICATION_CONTEXT);
```
Similarly this would create an attribute named attr1 in Request scope with value "Attr1 value".
```
pageContext.setAttribute("attr1", "Attr1 value", PageContext. REQUEST_CONTEXT);
```

pageContext Implicit Object Example

index.html

Here we are simply asking user to enter login details.
```
<html>
<head>
<title> User Login Page – Enter details</title>
</head>
<body>
<form action="validation.jsp">
Enter User-Id: <input type="text" name="uid"><br>
Enter Password: <input type="text" name="upass"><br>
<input type="submit" value="Login">
</form>
</body>
</html>
```

validation.jsp

In this page we are storing user's credentials using **pageContext** implicit object with the **session scope**, which means we will be able to access the details till the user's session is active. We can also store the attribute using other scope parameters such as page, application and request.

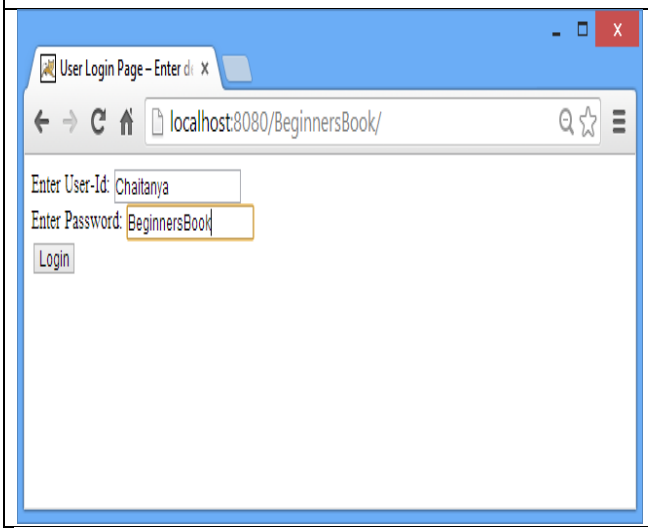
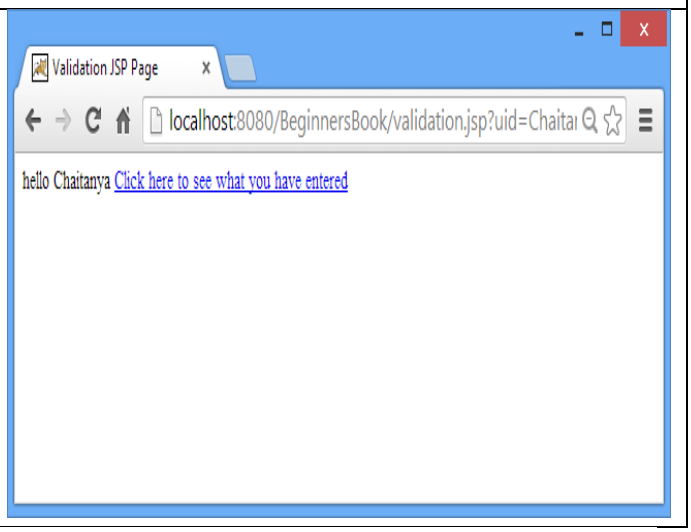
```
<html>
<head> <title> Validation JSP Page</title>
</head>
<body>
<%
String id=request.getParameter("uid");
String pass=request.getParameter("upass");
out.println("hello "+id);
pageContext.setAttribute("UName", id, PageContext.SESSION_SCOPE);
pageContext.setAttribute("UPassword", pass, PageContext.SESSION_SCOPE);
%>
<a href="display.jsp">Click here to see what you have entered </a>
</body>
</html>
```

display.jsp

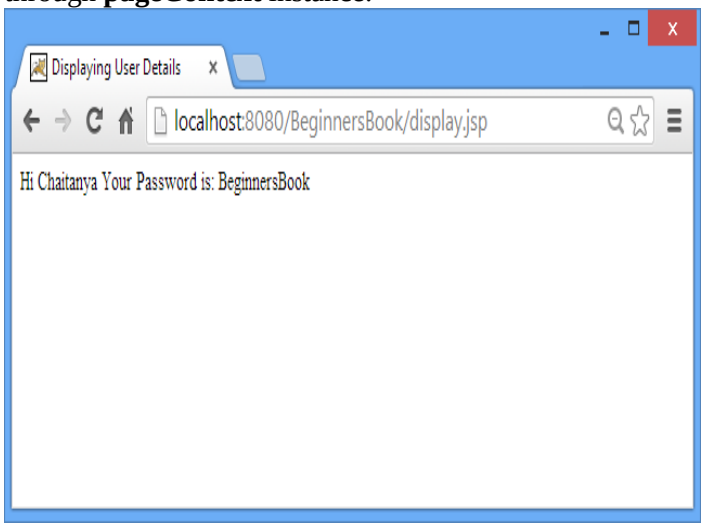
In this JSP page we are fetching the stored attributes using getAttribute method. The point to note here is that we have stored the attributes with session scope so we must need to specify scope as session in order to fetch those attribute's value.

```
<html>
<head>
<title>Displaying User Details</title>
</head>
<body>
<%
String username= (String) pageContext.getAttribute("UName", PageContext.SESSION_SCOPE);
String userpassword= (String) pageContext.getAttribute("UPassword", PageContext.SESSION_SCOPE);
out.println("Hi "+username);
out.println("Your Password is: "+userpassword);
%>
</body>
</html>
```

Screenshots of the example's output:

Login page where we are receiving User-Id and Password from user.	A page with details page link –
	

User Credentials display page which we have passed from login page to this page through **pageContext** instance.



JSP page implicit object

In JSP, page is an implicit object of type Object class.This object is assigned to the reference of auto generated servlet class. It is written as:

Object page=this;

For using this object it must be cast to Servlet type.For example:

<% (HttpServlet)page.log("message"); %>

Since, it is of type Object it is less used because you can use this object directly in jsp.For example:

<% this.log("message"); %>

JSP exception implicit object

It's an instance of java.lang.Throwable and frequently used for exception handling in JSP.
This object is only available for error pages, which means a JSP page should have isErrorPage to true in order to use exception implicit object.

The exception is normally an object that is thrown at runtime. Exception Handling is the process to handle the runtime errors. There may occur exception any time in your web application. So handling exceptions is a safer side for the web developer.

In JSP, there are two ways to perform exception handling:

1. By **errorPage** and **isErrorPage** attributes of page directive
2. By **<error-page>** element in web.xml file

Exception implicit Object Example

In this example we are taking two integer inputs from user and then we are performing division between them. We have used exception implicit object to handle any kind of exception in the below example.

index.html

```
<html>
<head>
<title>Enter two Integers for Division</title>
</head>
<body>
```

```
<form action="division.jsp">
Input First Integer:<input type="text" name="firstnum" />
Input Second Integer:<input type="text" name="secondnum" />
<input type="submit" value="Get Results"/>
</form>
</body>
</html>
```

Here we have specified exception.jsp as errorPage which means if any exception occurs in this JSP page, the control will immediately transferred to the exception.jsp JSP page.

Note: We have used **errorPage attribute of Page Directive** to specify the exception handling JSP page (<%@ page errorPage=”exception.jsp” %>).

division.jsp

```
<%@ page errorPage="exception.jsp" %>
<%
String num1=request.getParameter("firstnum");
String num2=request.getParameter("secondnum");
int v1= Integer.parseInt(num1);
int v2= Integer.parseInt(num2);
int res= v1/v2;
out.print("Output is: "+ res);
%>
```

In the below JSP page we have set **isErrorPage to true** which is also an attribute of Page directive, used for making a page eligible for exception handling.

Since this page is defined as a exception page in division.jsp, in case of any exception condition this page will be invoked. Here we are displaying the error message to the user using **exception implicit object**.

exception.jsp

```
<%@ page isErrorPage="true" %>
Got this Exception: <%= exception %>
Please correct the input data.
```

Output