**Types of Implicit Objects in JSP**

For every JSP file when the JSP compiler generates the servlet program it will create the following 9 implicit objects inside the \_jspService() method.

1. **Request**
2. **Response**
3. **PageContext**
4. **Session**
5. **Application**
6. **Config**
7. **Out**
8. **Page**
9. **Exception**

Hence all these objects are created and available inside the \_jspService() method we can use all these implicit objects directly inside the JSP scriptlets.

**Request Object in JSP**

It is an instance of **javax.servlet.http.HttpServletRequest** object. This implicit object is used to process the request sent by the client. It is the HttpServletRequest object associated with the request. When a client requests a page the JSP engine creates a new object to represent that request.

**Following are methods of request object:**

1. **getCookies()**: It returns an array containing all of the cookie objects the client sent with this request.
2. **getAttributeNames()**: It returns an enumeration containing the names of the attributes available to this request.
3. **getHeaderNames()**: It returns an enumeration of all the header names it contains.
4. **getParameterNames()**: It returns a String object containing the names of the parameters contained in this request.
5. **getSession()**: It returns the current session associated with this request.
6. **getLocale()**: It returns the preferred Locale that the client will accept content in.
7. **getInputStream()**: It helps us to retrieve the body of the request as binary data using a ServletInputstream.
8. **getAuthType()**: It returns the name of the authentication scheme used to protect the servlet.
9. **getContentType()**: It returns the MIME type of the body of the request.
10. **getContextPath()**: It returns the portion of the request URI.
11. **getMethod()**: It returns the name of the HTTP method with which this request was made.
12. **getPathInfo()**: It returns extra path information associated with the URL.
13. **getProtocol()**: It returns the name and version of the protocol the request uses.
14. **getQueryString()**: It returns the query string that is contained in the requested URL.
15. **getRequestedSessionId()**: It returns the session ID specified by the client.
16. **getServletPath()**: It returns the part of the request’s URL that calls the JSP.
17. **isSecure()**: It returns True or False indicating whether the request was made using a secure channel or not.
18. **getContentLength()**: It returns the length of the request body and made available by the input stream.
19. **getServerPort()**: It returns the port number on which the request was received.
20. **getRemoteAddr()**: It returns the IP address of the client that sent the request.

**Example of Request Object in JSP**

In this example, we are printing the name of the user with a welcome message. Here we are receiving the input from the user on the index.html page and displaying it in the welcome.jsp page using request implicit object.

**index.html**

<!DOCTYPE html>

**<html>**

**<head>**

**<title>**request implicit object**</title>**

**</head>**

**<body** bgcolor="pink"**>**

**<form** action="welcome.jsp"**>**

**<input** type="text" name="uname"**>** **<input** type="submit"

value="GO"**><br** **/>**

**</form>**

**</body>**

**</html>**

**welcome.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%

String name = request.getParameter("uname");

out.print("welcome " + name);

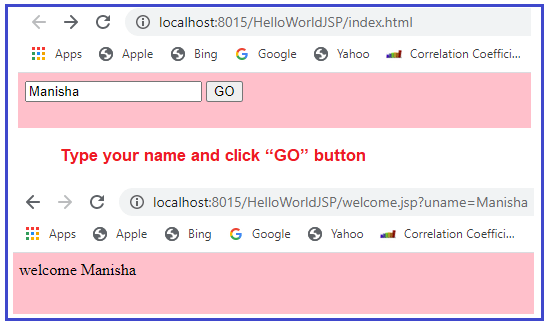
%>

**</body>**

**</html>**

**Output**

Run your code to get the following output.



**Response Object in JSP:**

This is the HttpServletResponse object associated with the response to the client. This implicit object is used to process the request and send the response to the client. We can send content-type as well as error reports and text. The object is by default available to scriptlets and expressions to get response-related information and to set response-related information. If a JSP wants to redirect a request from this server to another server then it can use the response object and sendRedirect method call.

**Following are the methods of response object:**

1. **encodeRedirectURL(String url)**: It is used to encode the specified URL for use in the sendRedirect method.
2. **containsHeader(String name)**: It returns True or False indicating whether the named response header has already been set.
3. **isCommitted()**: It returns True or False indicating if the response has been committed.
4. **addCookie()**: It is used to add the specified cookie to the response.
5. **addHeader(String name, String value)**: It is used to add a response header with the given name and value.
6. **flushBuffer()**: It is used to force any content in the buffer to be written to the client.
7. **reset()**: It is used to clear the data that exists in the buffer as well as the status code and headers.
8. **sendError(int sc)**: It is used to send an error response to the client using the specified status code and clearing the buffer.
9. **sendRedirect(String location)**: It is used to send a temporary redirect response to the client using the specified redirect location URL.
10. **setBufferSize(int size)**: It is used to set the preferred buffer size for the body of the response.
11. **setContentType(int len)**: It is used to set the length of the content body on the response.
12. **setContentType(String type)**: It is used to set the content type of the response being sent to the client.
13. **setHeader(String name, String value)**: It is used to set a response header with the given name and value.
14. **setLocale(Locale loc)**: It is used to set the locale of the response.
15. **setStatus(int sc)**: It is used to set the status code for the response.

**Example of Response Object in JSP**

In this example, we are redirecting the response to “Google”. Here we are receiving the input from the user on the index.html page and redirecting it on the welcome.jsp page using response implicit object.

**index.html**

<!DOCTYPE html>

**<html>**

**<head>**

**<title>**implicit object**</title>**

**</head>**

**<body** bgcolor="pink"**>**

**<form** action="welcome.jsp"**>**

**<input** type="text" name="uname"**>** **<input** type="submit"

value="GO"**><br** **/>**

**</form>**

**</body>**

**</html>**

**welcome.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%

response.sendRedirect("http://www.google.com");

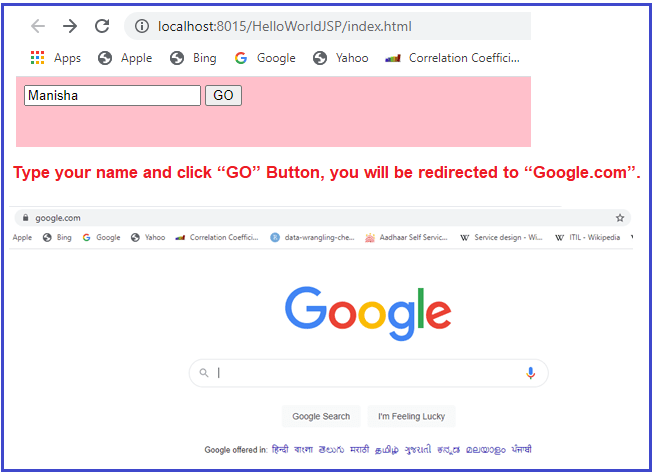
%>

**</body>**

**</html>**

**Output**

Run your code to get the following output:



**PageContext Implicit Object in JSP:**

pageContext is an implicit object in JSP Technology, it can be used to get all the JSP implicit objects even in a Non-JSP environment. It is created for pageContext class which is used for creating all the implicit variables like a session, ServletConfig, out, etc… While preparing Tag handler classes in custom tags container will provide pageContext object as part of its life cycle, from this we are able to get all the implicit objects. In JSP applications, by using the pageContext object we are able to perform some operations with the attributes in the JSP scopes page, request, session, and application like adding an attribute, removing an attribute, getting an attribute, and finding an attribute. This encapsulates the use of server-specific features like higher performance JSPWriters. The pageContext reference points to pageContext class sub-class object.

**Following are the methods of pageContext object:**

1. **setAttribute(String name, Object value, int scope)**: It is used to set an attribute onto a particular scope. Where scope may be page, request, session, or application.
2. **getAttribute(String name)**: It is used to get an attribute from page scope.
3. **getAttribute(String name, int scope)**: If we want to get an attribute value from the specified scope we have to use this method.
4. **removeAttribute(String name, int scope)**: It is used to remove an attribute from a particular scope.
5. **findAttribute(String name)**: To find an attribute value from page scope, request scope, session scope, and application scope we have to use this method.

**Example of PageContext Object in JSP:**

In this example, in index.html we are taking user input and storing user’s credentials using Pagecontext with the session scope is welcome.jsp page. So that we can able to access the details till the user’s session is active. Then we are fetching the stored attributes using the getAttribute method on the second.jsp page.

**index.html**

<!DOCTYPE html>

**<html>**

**<head>**

**<title>**implicit object**</title>**

**</head>**

**<body** bgcolor="pink"**>**

**<form** action="welcome.jsp"**>**

**<input** type="text" name="uname"**>** **<input** type="submit"

value="GO"**><br** **/>**

**</form>**

**</body>**

**</html>**

**welcome.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%

String name = request.getParameter("uname");

out.print("Welcome " + name);

pageContext.setAttribute("user", name, PageContext.SESSION\_SCOPE);

%>

**<a** href="second.jsp"**>**second jsp page**</a>**

**</body>**

**</html>**

**second.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%

String name = (String) pageContext.getAttribute("user", PageContext.SESSION\_SCOPE);

out.print("Hello " + name);

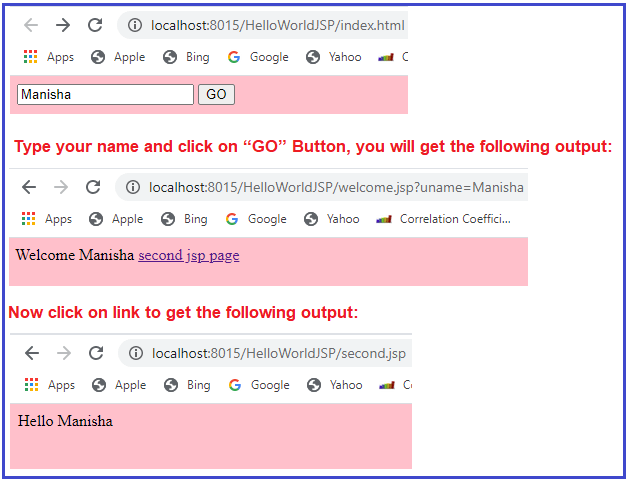
%>

**</body>**

**</html>**

**Output**

Run the index.html page, you will get the following output:



**Session Object in JSP:**

This is the HttpSession object associated with the request. a session is an implicit object which is created for HttpSession class using which we can store and access data. The session reference point HttpSession implementation object. By default, this reference is available to scriptlet, expression tags. These tags can access session related info and they can manage session scope attributes. By using this reference these tags can delete session objects by calling session.invalidate() and these reference tags can set session timeout.

**Following are the methods of the session object:**

1. **getAttribute(String name)**: It returns the object bound with the specified name in this session.
2. **getAttributeNames()**: It returns an Enumeration of String objects containing the names of all the objects bound to this session.
3. **getCreationTime()**: It returns the time when this session was created.
4. **getId()**: It returns a string containing the unique identifier assigned to this session.
5. **getLastAccessedTime()**: It returns the last time the client sent a request associated with this session.
6. **getMaxInactiveInterval()**: It returns the maximum time interval that the servlet container will keep this session open between client accesses.
7. **invalidate()**: It invalidates the session and unbinds any objects bound to it.
8. **isNew()**: It returns True if the client does not know about the session or if the client chooses not to join the session.
9. **removeAttribute(String name)**: It removes the object bound with the specified name from this session.
10. **setAttribute(String name, Object value)**: It binds an object to this session using the specified name.
11. **setMaxInactiveInterval(int interval)**: It specifies the time between client requests before the servlet container will invalidate this session.

**Example of Session Object in JSP:**

In this example, the index.html page would display a text box along with a submit button which would transfer the control to welcome.jsp page which will display the name user has entered. And it then stores the same variable in the session object so that it can be fetched on any page until the session becomes inactive. Then in second.jsp we are fetching the variable’s value from the session object and displaying it.

**index.html**

<!DOCTYPE html>

**<html>**

**<head>**

**<title>**implicit object**</title>**

**</head>**

**<body** bgcolor="pink"**>**

**<form** action="welcome.jsp"**>**

**<input** type="text" name="uname"**>** **<input** type="submit"

value="GO"**><br** **/>**

**</form>**

**</body>**

**</html>**

**welcome.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%

String name = request.getParameter("uname");

out.print("Welcome " + name);

session.setAttribute("user", name);

%>

**<a** href="second.jsp"**>**second jsp page**</a>**

**</body>**

**</html>**

**second.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%

String name = (String) session.getAttribute("user");

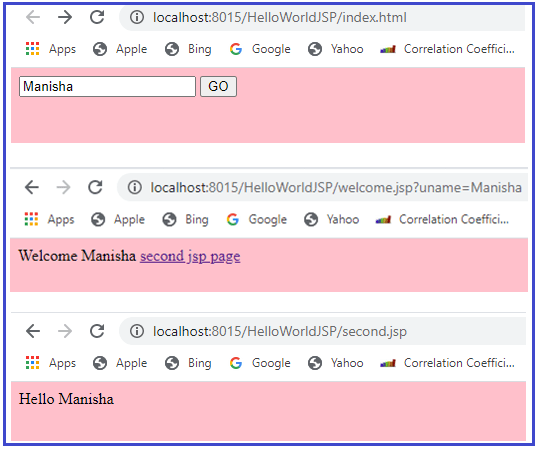
out.print("Hello " + name);

%>

**</body>**

**</html>**

**Output**



**Application Object in JSP**

This is the ServletContext object associated with the application context. Application is an implicit object which is created for ServletContext class and used to access the data of the ServletContaxt object same as in servlets. The application reference point ServletContext implementation object. By using this reference, a JSP can access application-related information such as managing application scope attributes, getting the application init parameters, etc.

**Following are the methods of application object:**

1. **setAttribute(String Key, Object Value)**: This method is used to set a hit counter variable and to reset the same variable.
2. **getAttribute(String Key)**: This method is used to read the current value of the hit counter and increase it by one and again set it for future use whenever a user accesses your page.

**Example of Implicit Application Object in JSP**

In this example, we are using the application object to get the initialization parameter from the web.xml configuration file.

**index.html**

<!DOCTYPE html>

**<html>**

**<head>**

**<title>**implicit object**</title>**

**</head>**

**<body** bgcolor="pink"**>**

**<form** action="welcome"**>**

**<input** type="text" name="uname"**>** **<input** type="submit"

value="GO"**><br** **/>**

**</form>**

**</body>**

**</html>**

**welcome.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%

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

String driver = application.getInitParameter("dname");

out.print("driver name is=" + driver);

%>

**</body>**

**</html>**

**web.xml**

**<?xml version="1.0" encoding="UTF-8"?>**

**<web-app** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"

id="WebApp\_ID" version="4.0"**>**

**<servlet>**

**<servlet-name>**welcome**</servlet-name>**

**<jsp-file>**/welcome.jsp**</jsp-file>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>**welcome**</servlet-name>**

**<url-pattern>**/welcome**</url-pattern>**

**</servlet-mapping>**

**<context-param>**

**<param-name>**dname**</param-name>**

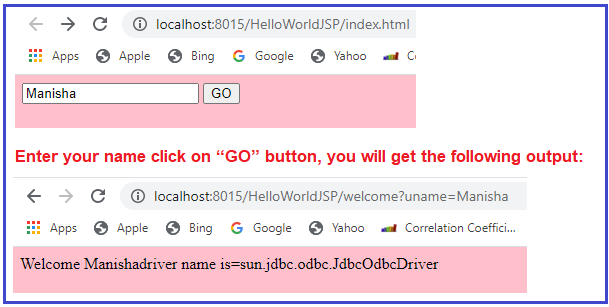
**<param-value>**sun.jdbc.odbc.JdbcOdbcDriver**</param-value>**

**</context-param>**

**</web-app>**

**Output**

Run your code to get the following output:



**Implicit Config Object in JSP:**

This is the ServletContext object associated with the page. This implicit object is created for ServletConfig class and used to access the data of the ServletConfig object same as in servlets. The config reference points ServletConfig implementation object. By using this config reference a JSP page can get initialization parameters of JSP configuration, get initialization parameters names, get servlet names, and can get servlet context reference.

**Following is the only method of config object:**

1. **getServletName()**: This method returns the servlet name which is the string contained in the <servlet-name> element defined in the web.xml configuration file.
2. **getServletContext()**: It returns a reference to the Servlet Context.

**Example of Config Object in JSP**

In this example, we are calling the getServletName() method of config object for fetching the servlet name from web.xml.

**index.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%

String sname = config.getServletName();

out.print("Servlet Name is: " + sname);

%>

**</body>**

**</html>**

**web.xml**

**<?xml version="1.0" encoding="UTF-8"?>**

**<web-app** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"

id="WebApp\_ID" version="3.0"**>**

**<servlet>**

**<servlet-name>**WelcomeServlet**</servlet-name>**

**<jsp-file>**/index.jsp**</jsp-file>**

**</servlet>**

**<servlet-mapping>**

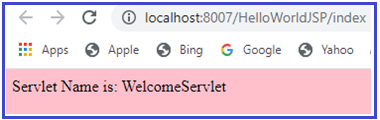
**<servlet-name>**WelcomeServlet**</servlet-name>**

**<url-pattern>**/index**</url-pattern>**

**</servlet-mapping>**

**</web-app>**

**Output**



**Out Object in JSP:**

This is the PrintWriter object used to send output to the client. The out-reference points JSPWriter subclass object. By using the object, we can print HTML tags and plain text data on the browser.

**Methods of out object**

1. **out.print(dataType dt)**: It is used to print a data type value.
2. **out.println(dataType dt)**: It is used to print a data type value then terminate the line with a new line character.
3. **out.flush()**: It is used to flush the stream.

**Example of Out Object in JSP**

In this example, we are using the print method of OUT for displaying messages to the client.

**index.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%

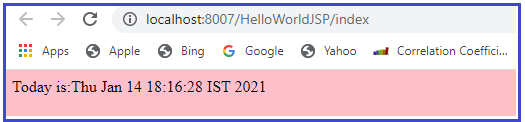
out.print("Today is:" + java.util.Calendar.getInstance().getTime());

%>

**</body>**

**</html>**

**Output**



**Page Object in JSP**

This is simply a synonym for this and is used to call the methods defined by the translated servlet class. The page reference points current JSP page object.

**Example**

**public final view\_jsp extends HttpBase…{**  
**public void \_jspService(request, response)…{**  
**Object page=this;**  
**}**  
**}**

**Exception Object in JSP**

The Exception object allows the exception data to be accessed by a designated JSP. This reference points to any exception object raised in the .jsp page and provides that exception object to the error JSP page.

**Methods of Exception implicit object**

1. **getMessage()**: It gives a detailed message about the exception that has occurred. This is initialized in the Throwable constructor.
2. **getCause()**: It returns the cause of the exception as represented by a Throwable object.
3. **toString()**: It gives the name of the class concatenated with the result of getMessage().
4. **printstackTrace()**: It prints the result of toString() along with the stack trace to System.err.
5. **getStackTrace()**: It returns an array containing each element on the stack trace.
6. **fillnStackTrace()**: It fills the stack trace of this Throwable object with the current stack trace.

**Example of Implicit Exception Object in JSP**

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

**index.html**

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

**<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>**

**division.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%@ 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);

%>

**</body>**

**</html>**

**exception.jsp**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Insert title here**</title>**

**</head>**

**<body** bgcolor="pink"**>**

<%@ page isErrorPage="true"%>

Got this Exception:

<%=exception%>

Please correct the input data.

**</body>**

**</html>**

**Output**