Web Component Development Using Java

Session: 8

JSP Implicit Objects

Objectives

- Explain the concept of implicit objects in JSP
- List various types of implicit objects in JSP
- Explain how to use the request object
- Explain how to use the response object
- Explain how to use the out object
- Explain how to use the session object
- Explain how to use the application object
- Explain how to use the pageContext object
- Explain how to use the page object
- Explain how to use the config object
- Explain how to use the exception object

Implicit Objects

Implicit objects:

- Are a set of Java objects that are available in every JSP page.
- Are created and loaded by the Web container.
- Are referred to as pre-defined variables.
- Are accessible within the scripting elements in the JSP page.

Where are implicit objects created?

- When the JSP page is translated into a Servlet class, all the implicit objects declarations are taken within the `_jspService()` method.

Types of Implicit Objects 1-2

- Some of the implicit objects provided by JSP are classified into following categories:
  - Input and Output Objects
  - Scope Communication and Control Objects
  - Servlet Objects
  - Error Objects

Input and Output Objects

- These include objects such as `request`, `response`, and `out`.
- The `request` object controls the data coming into the page.
- The `response` object controls the information generated as a result of request object.
- The `out` object controls the output data generated as a response to the request.

Types of Implicit Objects 2-2

Scope Communication and Control Objects

- The scope communication objects provide access to all the objects available in the given scope.
- Objects in a JSP application are accessible according to the specified scope.
- The scope of an object is the section where that object is accessible.

Servlet Objects

- These objects provide information about the page context.
- These processes request object from a client and sends the response object back to the client.

Error Objects

- The error object handles error in a JSP page using an implicit object known as `Exception`.
- User can access this object by declaring JSP page as an error page.
- To do so, use the `isErrorPage` attribute of the page directive. For example, `<%@page isErrorPage="true" %>`.

Input and Output Objects 1-9

request Object

- It contains information sent from the client browser through HTTP protocol.
- The information stored in the request object includes source of the request, requested URL headers, cookies, and parameters associated with the request.
- When the JSP page is translated into the Servlet class, the request object is passed as a reference of the `javax.servlet.http.HttpServletRequest` interface by the container.
- The scope of the request object is page-level which means that the object will be accessible only in the current page.

Input and Output Objects 2-9

Table lists some of the methods of the `request` object available on the JSP page for the developers.

Method	Description
`java.lang.String getParameter(java.lang.String name)`	This method returns the value of a request parameter as a string, or `null` if the parameter does not exist.
`java.lang.String[] getParameterValues(java.lang.String name)`	This method returns an array of string objects which contains all the values the given request parameter has, or `null` if the parameter does not exist.
`java.util.Enumeration getParameterNames()`	This method returns an enumeration of string objects, which contains the names of the parameters. If no parameters, this method returns an empty enumeration.
`void setAttribute(java.lang.String name,java.lang.Object o)`	This method stores an attribute in the request. Attributes are reset between requests.
`java.lang.Object getAttribute(java.lang.String name)`	This method returns the name of the string from the named attribute.
`java.lang.String getRemoteHost()`	This method returns the name of the client or the last proxy that sends the request. If the hostname is not resolved, this method returns the dotted-string form of the IP address.
`java.lang.String getRemoteAddr()`	This method returns the Internet protocol address of the client or last proxy that sent the request.

Input and Output Objects 3-9

The code snippet demonstrates the methods of `request` object to receive data from a registration page.

```
<!-- index.jsp -->

. . .
<form action="register.jsp">
 First Name:
 <input type="text" name="firstname">

 <br/> <br/>

 Favorite Game:
 <input type="checkbox" name="game" value="Cricket">Cricket<BR>

 <input type="checkbox" name="game" value="Hockey">Hockey<BR>

 <input type="checkbox" name="game"
value="Baseball">Baseball<BR>

 <input type="checkbox" name="game"
value="Badminton">Badminton<BR>

<br/> <br/>
 <input type="submit" value="Submit" name="Submit">
 </form>
. . .
```

Input and Output Objects 4-9

```
<!-- register.jsp →

. . .
 <%
            // Retrieves data from text field
            String firstName = request.getParameter("firstname");

            // Retrieves data from check boxes and list box
            String favGame[] = request.getParameterValues("game");

            // Displays the form data
            out.println("Name is: " + "<font color='blue'>" + firstName +
"</font>");

            out.println("<br/><br/>Favorite Game: " + "\n");
            for (int i = 0; i < favGame.length; i++) {
                out.println("\t<font color='blue'>" + favGame[i] +
"</font>");
            }

 %>
      <br/> <br/> The parameters sent to the Servlet are: <br/>

      <font color="blue">
      <%
        Enumeration enumParam = request.getParameterNames();
        while (enumParam.hasMoreElements()) {
              String str = (String) enumParam.nextElement();
              out.println(str);
            }
      %>
      </font>
. . .
```

Input and Output Objects 5-9

response Object

- Manages the response generated by JSP and uses HTTP protocol to send the response to client.
- Is a reference of the `javax.servlet.http.HttpServletResponse` interface.

Table lists the methods of `response` object to be used in the JSP page.

Method	Description
`public void addCookie(Cookie cookie)`	The method adds the specified cookie to the response. This method can be called more than once to set more than one cookies.
`public void sendRedirect(java.lang.String location)throws java.io.IOException)`	The method sends a temporary redirect response to the client using the specified redirect location URL. This method can also accept relative URLs.
`public java.lang.String encodeURL(java.lang.String url)`	The method is used to encode the specified URL by including session id. This form of the url can be used in html tags that use a url, such as **<a href.....>**. This enables the server to keep track of the session.
`public java.lang.String encodeRedirectUrl(java.lang.String url)`	The method returns a rewritten url that can be used with the **sendRedirect** method of **response** object.

Input and Output Objects 6-9

out Object

- Represents the output stream, which will be sent to the client as a response for the request.
- Is an instance of the `javax.servlet.jsp.JspWriter` class.
- Uses all standard `write()`, `print()`, and `println()` methods defined in `javax.servlet.jsp.JspWriter` class to display the output.
- Has page scope.

Input and Output Objects 7-9

- Some of the methods of `out` object are as follows:
  - **`void flush()throws java.io.IOException`**
    - Flushes the buffer.
    - Only the invocation of `flush()` will flush all the buffers in a chain of writers and output streams.
    - May be invoked indirectly if the buffer size is more.

  The code snippet show how to flush the page content.

```
 out.flush();
<jsp:forward page="login.jsp"/>
```

  `void clear()throws java.io.IOException`
    - Clears the contents of the buffer.
    - If the buffer has been already been flushed, then the clear operation will throw an `IOException`.

Input and Output Objects 8-9

The code snippet shows how to clear the buffer.

```
// If buffer is not empty, buffer is cleared
   if (out.getBufferSize() != 0)
       out.clear();
```

- **`void close()throws java.io.IOException`** - For the `JspWriter`, this method need not be invoked explicitly as the code generated by the JSP container will automatically include `close()`

The code snippet shows the use of `close()` method.

```
out.print("Welcome!!!");
out.close();
```

- **`void println(java.lang.String x)throws java.io.IOException`** - This method prints a string and then terminates the line. For example, `out.println("The output is:" + ex);`

Input and Output Objects 9-9

- **`void print(java.lang.Strings)throws java.io.IOException`** - This method prints a string. If the argument is `null`, then the string `'null'` is printed.

The code snippet demonstrates the use of `print()` method.

```
out.print("Welcome!!!");
out.print("To Aptech Training");
```

Scope Communication Objects 1-13

- The scope communication objects include:

- **Session Object:**
  - Is an instance of `javax.servlet.http.HttpSession` interface.
  - Has session scope.

- Some of the methods of **session** object are:
  - **`public boolean isNew()`** - The `isNew()` method is used for checking whether the session is newly created in this page or not. It will return false, if a session already exists and true, otherwise.

Scope Communication Objects 2-13

- **`public void setAttribute(java.lang.String name,java.lang.Object value)`** - This method holds the objects in the existing session.

The code snippet demonstrates how to associate a new session object with the request.

```
. . .
<%
 if (session.isNew())
 {
   UserLogin userLoginObj = new UserLogin(name,
password);

   session.setAttribute("loginuser", userLoginObj);
 }
%>
. . .
```

Scope Communication Objects 3-13

- **public java.lang.Object getAttribute(java.lang.String name)** - The method returns the object bound with the specified name in the current session, otherwise returns null.

The code snippet shows how to access the named `session` object.

```
. . .
<%
String name = (String)
session.getAttribute("loginuser");
%>
<table border="1">
 <tr>
     <td> User name: </td>
     <td> <% out.print(name); %> </td>
 </tr>
</table>
. . .
```

Scope Communication Objects 4-13

- **public java.util.Enumeration getAttributeNames()** - The method returns an enumeration of string objects. The string contains the names of all the objects bound to this session.

- **public void invalidate()** - The method invalidates a session and then, releases the objects bound to it.

The code snippet shows how to invalidate a session.

```
<%
// Retrieve the user's session
   HttpSession session = request.getSession();

// Invalidate the session
   session.invalidate();
%>
```

- **public void removeAttribute(java.lang.String name)** - The method removes the object bounded with the specified name from this session. For example,
`session.removeAttribute("loginuser");`

Scope Communication Objects 5-13

- **application Object:**
  - Is used to share the data between all the application pages.
  - Can be accessed by any JSP present in the application.
  - Is an instance of the `javax.servlet.ServletContext` interface.
  - Has application scope.

  Some of the methods of `application` object are:

  - **`public void setAttribute(java.lang.String name, java.lang.Object object)`** - The method returns the servlet container attribute with the given name or null, if there is no attribute.

  The code snippet shows how to set an attribute to be accessed in any JSP page in the application.

```
. . .
<%
   String Initialized = "yes";
   application.setAttribute("init", Initialized);
%>
```

Scope Communication Objects 6-13

- `public java.lang.Object getAttribute(java.lang.String name)` - The method returns the servlet container attribute with the given name or null, if there is no attribute.
- `public java.util.Enumeration getAttributeNames()` - The method returns an enumeration containing the attribute names available within this servlet context.

Scope Communication Objects 7-13

The code snippet demonstrates how to retrieve all application objects set in the application in the JSP page.

```
<%
  String username="john";
  String password="apt001";

    application.setAttribute("username", username);
    application.setAttribute("password", password);

// Retrieves all attributes
    Enumeration enum=application.getAttributeNames();
// Print the attributes
    while(enum.hasMoreElements()){
       String attrName=(String)enum.nextElement();
       out.println(attrName+"<BR>");
    }
%>
```

Scope Communication Objects 8-13

- **`public void removeAttribute(java.lang.String name)`** - This method removes the attribute with the given name from the servlet context. For example,
  `application.removeAttribute("password");`

- **`public java.lang.String getServerInfo()`** - This method returns the name and version of the servlet container on which the servlet is running. For example, `out.println("Server Information:"+ application.getServerInfo());`

- **`public void log(java.lang.String msg)`** - This method writes the specified message to a servlet log file, usually an event log. For example, `application.log (Hello log Message);`

Scope Communication Objects 9-13

- **pageContext Object:**
  - Allows user to access all the implicit objects defined in the page scope.
  - Is an instance of the `javax.servlet.jsp.PageContext` class.
  - Uses methods defined in the `PageContext` class to access implicit objects in a Web page.
- `PageContext` class provides following fields that are used by `pageContext` object to find the scope or specify the scope of the objects:

PAGE_SCOPE	• Specifies the scope for attributes stored in the `pageContext` object. This is the default.
REQUEST_SCOPE	• Specifies the request scope for attributes that remain available until current request is complete.
SESSION_SCOPE	• Specifies the scope for attributes stored in the `session` object.
APPLICATION_SCOPE	• Specifies the scope for attributes stored in the `application` object that remains available until it is reclaimed.

Scope Communication Objects 10-13

- The `pageContext` object provides the following methods to access all the attributes defined by implicit object in a page:
  - **`void setAttribute(java.lang.String name, java.lang.Object value, int scope)`** - This method registers the name and value specified with appropriate scope. If the value passed in is `null`, this method functions in the same way as `removeAttribute(name, scope)`.
  - **java.lang.Object getAttribute(java.lang.String name)** - This method returns the object associated with the name in the page scope, otherwise it returns null. It throws `java.lang.NullPointerException`, if the name is null.

Scope Communication Objects 11-13

The code snippet shows the `pageContext` object used to set and retrieve the object.

```
<%
String personName = "Cleveland";
pageContext.setAttribute("name", personName,
APPLICATION_SCOPE);
%>
. . .

<%
  String personName = (String)
pageContext.getAttribute("name", APPLICATION_SCOPE);
  out.print("Person name:" + personName);
%>
```

- **`public abstract java.util.Enumeration getAttributeNamesInScope(int scope)`** - This method enumerates all the attributes in a given scope. It throws `java.lang.IllegalArgumentException` if the scope is invalid.

Scope Communication Objects 12-13

The code snippet demonstrates how to retrieve all the objects with request scope.

```
// Prints all attribute names with request scope
<%
   for (java.util.Enumeration e =

pageContext.getAttributeNamesInScope(javax.servlet.j
sp.PageContext.REQUEST_SCOPE) ;

   e.hasMoreElements(); ) { %>

<%= e.nextElement() %><br>

<%  } %>
```

- **public abstract void removeAttribute(java.lang.String name)** - This method removes the object reference associated with the given name from all the scopes. It throws `java.lang.NullPointerException`, if the name is null.

Scope Communication Objects 13-13

The code snippet demonstrates how to remove the objects from the JSP page.

```
<%
out.println("User Name: " +
request.getAttribute("Name") + "<br/>");

out.println("Password: " +
request.getAttribute("Password") + "<br/>");

pageContext.removeAttribute("Password");
%>
```

Servlet Objects 1-5

- Include objects:

- **page Object:**
  - It is an instance of the `java.lang.Object`.
  - It is an instance of the servlet processing the current request in a JSP page.
  - It has page scope.

Servlet Objects 2-5

The code snippet demonstrates how to access the methods of the Servlet through page object.

```
<%
      out.println(this.getServletInfo());
      out.println(((Servlet)page).getServletInfo());
 %>
```

config Object:

- Stores the information of the servlet, which is created during the compilation of a JSP page.
- Is an instance of the `javax.servlet.ServletConfig` interface.
- Provides methods to retrieve servlet initialization parameters.
- Represents the configuration of the servlet data where a JSP page is complied.
- Has page scope.

Servlet Objects 3-5

- Some of the methods of config object are:

```
public java.lang.String getInitParameter(java.lang.String name)
```

- This method returns a string which contains the value of the named initialization parameter, otherwise it returns `null`.

```
public java.util.Enumeration getInitParameterNames()
```

- Returns the names of the servlet's initialization parameters as an enumeration of string objects.

Servlet Objects 4-5

- Code Snippet first retrieves all the init parameters `getInitParameterNames()` and then, prints each parameter name using `getInitParameter()`.

```
public void getParameters(ServletConfig config)
throws ServletException {

    // Retrieves all init parameters
     Enumeration params =
config.getInitParameterNames();

     // Prints the init parameter names and values
     while (params.hasMoreElements()) {
       String name = (String) params.nextElement();
       System.out.println("Parameter Name: " + name +
" Value: " + config.getInitParameter(name));
   }
}
```

Servlet Objects 5-5

public ServletContext getServletContext()

- This method returns a reference to the servlet context.
- The code snippet given shows how to set a JavaBean object in the `ServletContext`.

```
ServletContext context =
config.getServletContext();
context.setAttribute("dbBean", dbBean);
```

public java.lang.String getServletName()

- This method returns the name of this servlet instance.
- It can get the name that may be provided through the server administration or assigned in the Web application deployment descriptor.
- For example, `String servletName = config.getServletName();`

exception Object 1-2

- Runtime errors can be processed using the `exception` object present in the JSP page.
- The `exception` object:
  - Is used to trace the exception thrown during execution.
  - Is available to the JSP page that is assigned as an error page.
  - Is the instance of `java.lang.Throwable` class.
  - Has page scope.
- Some of the methods of `exception` object are:

`public String getMessage()`

- Returns the descriptive error message associated with the exception when it was thrown.

`public void printStackTrace(PrintWriters)`

- Prints the execution stack in effect when the exception was thrown to the designated output stream.

exception Object 2-2

public String toString()

- Returns a string combining the class name of the exception with its error message.

- Code Snippet shown demonstrates how to handle exceptions in the JSP.

```
<%@ page isErrorPage="true" %>
<html>
<head>
    <title>Implicit Object</title>
</head>
<body>
    <h1>Implicit Object: Exception</h1>
    The following error has been detected :<br>
    <b><%= exception %></b><br>
    <% exception.printStackTrace(out); %>
</body>
</html>
```

  - The `isErrorPage` attribute is set to true, which indicates that the page is an error page. The JSP expression (`<%= %>`) is used to create an instance of the `exception` object.
  - The `out` object is passed as an argument to the `printStackTrace()` method which will display the stack trace information.

Summary

- JSP implicit objects are a set of Java objects that are created and maintained automatically by the Web container.
- The request object represents the request from the client for a Web page.
- The response implicit object handles the response generated by JSP and sends response to the client.
- The out object represents the output stream.
- The Web server creates a session object for multiple requests sent by a single user. The application object represents the application to which the JSP page belongs.
- The pageContext object allows user to access all the implicit objects defined in the page scope.
- The page object provides access to all the objects, which are defined in a Web page.
- The config object stores the information of the servlet.
- When an error occurs, the execution of a JSP page is terminated. The exception implicit object is used to trace exceptions that occur in a JSP page.