JSP Introduction Quiz

1. What is the main value of Java Server Pages?

JSP separate display from processing, i.e., separate html from java code.

2. How are JSP pages related to servlets?

Server converts the JSP into a servlet class. The JSP file is compiled only when the page is first requested, or when the JSP file has been updated and has a more recent timestamp.

3. How are JSP pages related to HTML?

Java Server Pages (JSP) are a technology for developing web pages that support dynamic content, allowing developers to insert java code in HTML pages by making use of special JSP tags.

4. Give an example of a JSP scriptlet and show how it will look in the JSP servlet.

```
<html>
      <body>
            <% int count=0; %>
            The page count is now:
            <%= ++count %>
      </body>
</html>
```

How it will look in the JSP servlet:

```
public class basicCounter_jsp extends SomeSpecialHttpServlet {
      public void _jspService(HttpServletRequest request,
      HttpServletResponse response) throws java.io.IOException,
      ServletException {
            PrintWriter out = response.getWriter();
            response.setContentType(“text/html”);
            out.write(“<html><body>”);
            int count=0;
            out.write(“The page count is now:”);
            out.print(++count);
            out.write(“</body></html>”);
      }
}
```

5. Give an example of a JSP declaration and show how it will look in the JSP servlet.

```
<html>
      <body>
            <%! int count=0; %>
            The page count is now:
            <%= ++count %>
      </body>
</html>
```

How it will look in the JSP servlet:

```
public class basicCounter_jsp extends SomeSpecialHttpServlet {
      int count=0;

      public void _jspService(HttpServletRequest request,
      HttpServletResponse response) throws java.io.IOException,
      ServletException {
            PrintWriter out = response.getWriter();
            response.setContentType(“text/html”);
            out.write(“<html><body>”);
            out.write(“The page count is now:”);
            out.print(++count);
            out.write(“</body></html>”);
      }
}
```

6. Give an example of a JSP comment and show how it will look in the JSP servlet.

```
<html>
      <body>
            <%-- This is a comment --%>
            Hello world
      </body>
</html>
```

How it will look in the JSP servlet:

```
public class basic_jsp extends SomeSpecialHttpServlet {
      public void _jspService(HttpServletRequest request,
      HttpServletResponse response) throws java.io.IOException,
      ServletException {
            PrintWriter out = response.getWriter();
            response.setContentType(“text/html”);
            out.write(“<html><body>”);
            out.write(“<!-- This is a comment -->”);
            out.write(“Hello world”);
            out.write(“</body></html>”);
      }
}
```

7. Give an example of a JSP expression and show how it will look in the JSP servlet.

```
<html>
      <body>
            <% int count=0; %>
            The page count is now:
            <%= ++count %>
      </body>
</html>
```

How it will look in the JSP servlet:

```
public class basicCounter_jsp extends SomeSpecialHttpServlet {
      public void _jspService(HttpServletRequest request,
      HttpServletResponse response) throws java.io.IOException,
      ServletException {
            PrintWriter out = response.getWriter();
            response.setContentType(“text/html”);
            out.write(“<html><body>”);
            int count=0;
```

```
                out.write(“The page count is now:”);
                out.print(++count);
                out.write(“</body></html>”);
        }

}
```

8. Give an example of a JSP directive and show how it will look in the JSP servlet.

```
<%@ page import="java.util.Date" %>
<html>
        <body>
        <% Date date = new Date(); %> Hello! The time is now <%= date %>
        </body>
</html>
```

How it will look in the JSP servlet:

```
import java.util.Date;
public class basic_jsp extends SomeSpecialHttpServlet {
        public void _jspService(HttpServletRequest request,
        HttpServletResponse response) throws java.io.IOException,
        ServletException {
                PrintWriter out = response.getWriter();
                response.setContentType(“text/html”);
                out.write(“<html><body>”);
                Date date = new Date();
                out.write(“Hello! The time is now ” + date);
                out.write(“</body></html>”);
        }
}
```

9. Explain how an EL expression is evaluated.

${something}

- container evaluates this as follows
- checks page scope for an attribute named "something", if found use it.
- otherwise check request scope for an attribute named "something", if found use it.
- otherwise check session scope for an attribute named "something", if found use it
- otherwise check application scope for an attribute named "something", if found use it.
- otherwise ignore the expression.

${firstThing}

- if firstThing is not an implicit EL object, then search page, request, session and application scopes until attribute "firstThing" is found.

${firstThing.secondThing}

- if firstThing is a bean then secondThing is a property of the bean.

- if firstThing is a map then secondThing is a key of the map.

${firstThing[secondThing]}

- if firstThing is a bean then secondThing is a property of the bean.
- if firstThing is a map then secondThing is a key of the map.
- if firstThing is a List then secondThing is an index into the List.

10. Explain how servlet attributes are involved in EL expressions.

When EL expressions evaluate attribute, they will find in the implicit objects such as pageScope, sessionScope, requestScope…

11. Explain how servlets and JSPs use request dispatch to interact.

- Can invoke a servlet from a JSP page through the `jsp:include` and `jsp:forward` action tags and can pass data from a JSP page to servlet with `jsp:param` tag is used within those tags.
- Can invoke a JSP page from a servlet through functionality of the standard `javax.servlet.RequestDispatcher` interface. And invoke the `include()` or `forward()` method of the request dispatcher, specifying the HTTP request and response objects as arguments.