JSP

- JSP technology is used to create web application just like servlet technology.
- It can be thought of as an extension to servlet because it provides **more functionality than servlet such as expression language, jstl**, etc.
- A jsp page consists of html tags and jsp tags.
- The jsp pages are easier to maintain than servlet because we can separate designing and development.
- It provides some additional features such as expression language, custom tags, etc.

ADVANTAGES OF JSP OVER SERVLET

Extension to servlet

JSP uses all the features of the servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and custom tags in JSP, that makes JSP development easy.

- Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic.

- Fast development: no need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project.

- Less code than servlet:

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code.

JSP REQUEST-RESPONSE CYCLE

- A JSP life cycle is defined as the process from its creation till the destruction.
- This is similar like a servlet life cycle with an additional step which is required to compile a jsp into servlet.
- Following paths followed by JSP:
    - Compilation
    - Initialization
    - Execution
    - Cleanup

JSP REQUEST-RESPONSE CYCLE

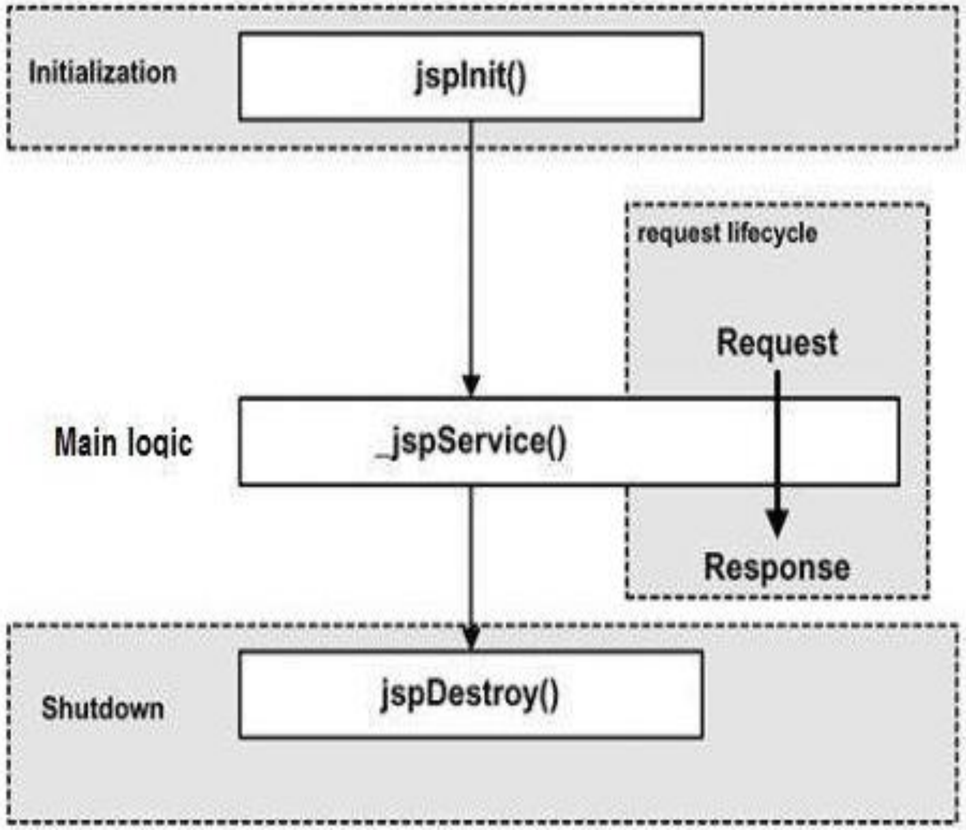
1)JSP initialization

- When a container loads a JSP it invokes the jspinit() method before servicing any requests. If you need to perform jsp-specific initialization, override the jspinit() method –
- public void jspInit(){

  // Initialization code...

}

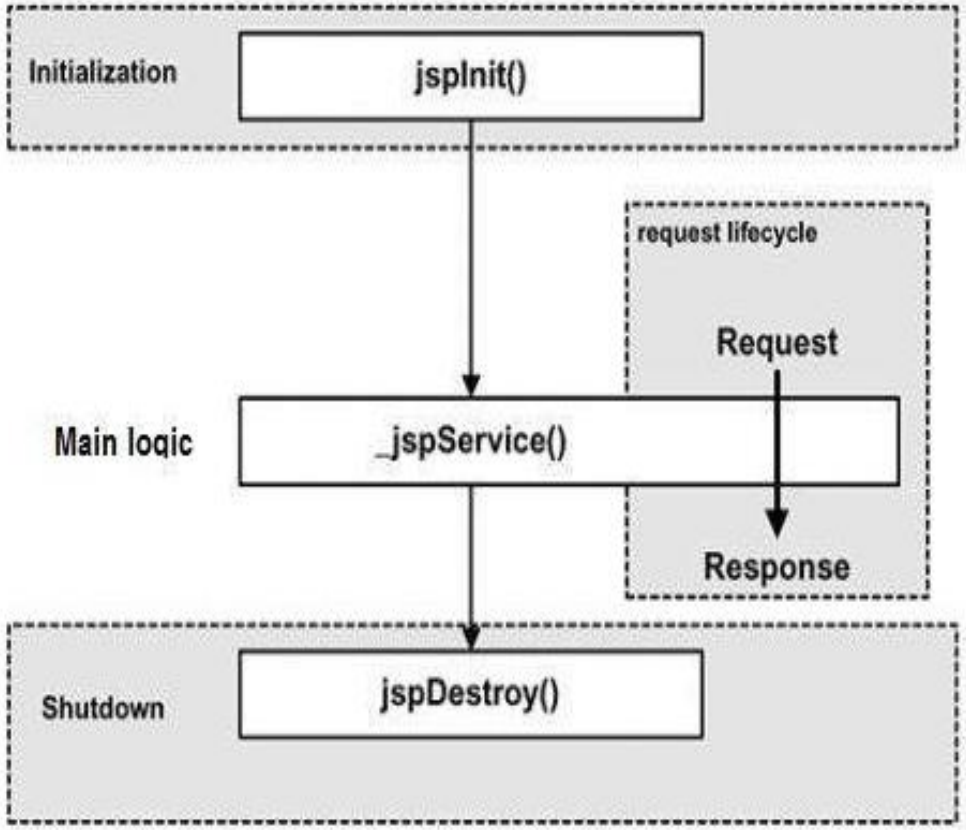
- you generally initialize database connections, open files, and create lookup tables in the jspInit method.



JSP REQUEST-RESPONSE CYCLE

JSP Execution

- Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the _jspService() method in the JSP.
- The _jspService() method takes an HttpServletRequest and an HttpServletResponse as its parameters as follows −
- The _jspService() method of a JSP is invoked on request basis.
- This is responsible for generating the response for that request and this method is also responsible for generating responses to HTTP methods like GET ,POST etc.



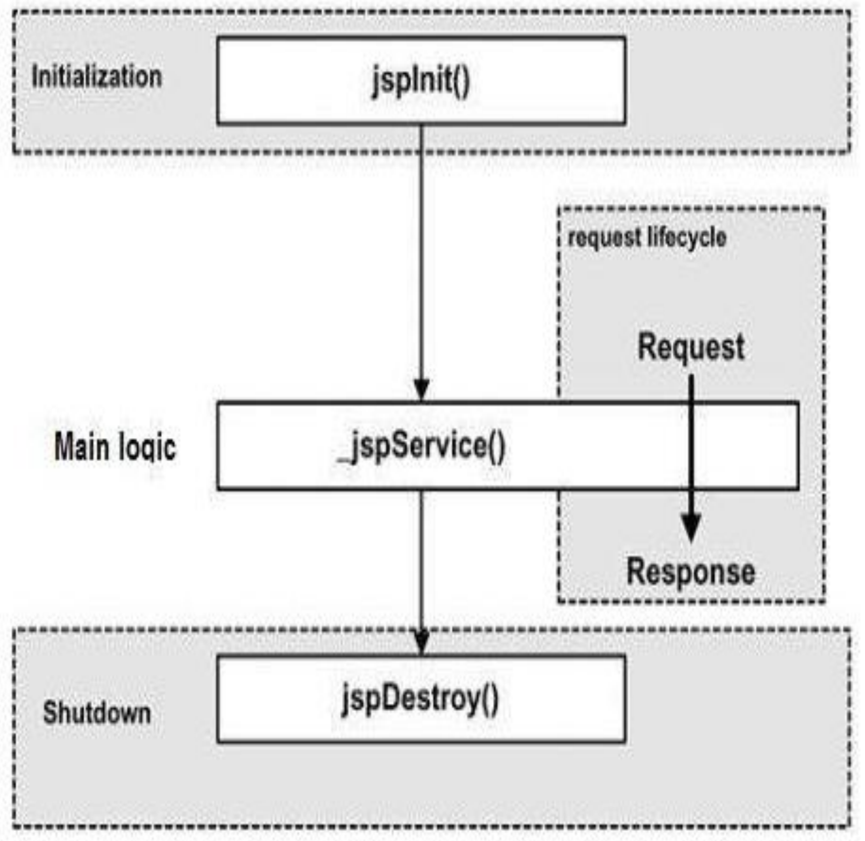
JSP REQUEST-RESPONSE CYCLE

JSP cleanup-

- The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container.
- The **jspdestroy()** method is the jsp equivalent of the destroy method for servlets.
- Override **jspdestroy()** when you need to perform any cleanup, such as releasing database connections or closing open files.

The jspDestroy() method has the following form −

```
public void jspDestroy() {
  // Your cleanup code goes here.
}
```



Syntax in JSP-Elements of JSP

1) The scriptlet:

- A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.
- Following is the syntax of Scriptlet −

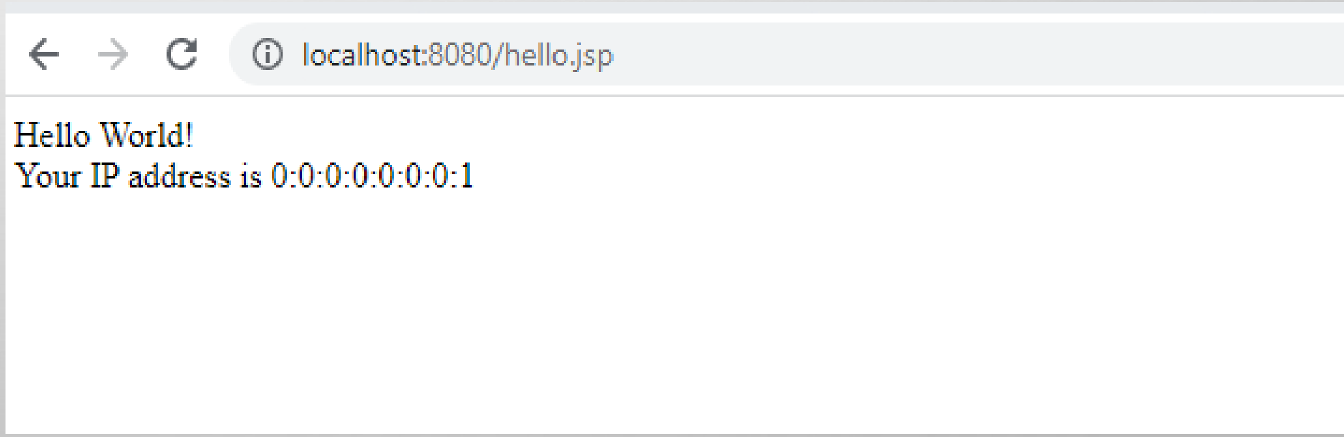
```
<% code fragment %>
```

- Any text, HTML tags, or JSP elements you write must be outside the scriptlet. Following is the simple and first example for JSP −

First example of JSP

```
<html>
  <head><title>Hello World</title></head>
  <body>
    Hello World!<br/>
    <%
      out.println("Your IP address is " + request.getRemoteAddr());
    %>
  </body>
</html>
```

1. Save as the file as say hello.jsp and save it in C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps\ROOT
2. Browse through the same using URL http://localhost:8080/hello.jsp



localhost:8080/hello.jsp

```
Hello World!
Your IP address is 0:0:0:0:0:0:0:1
```

JSP DECLARATIONS

- A declaration declares one or more variables or methods that you can use in java code later in the JSP file. You must declare the variable or method before you use it in the JSP file.
- Following is the syntax for JSP Declarations −

```
<%! declaration; [ declaration; ]+ ... %>
```

- For ex:

```
<%! int i = 0; %>

<%! int a, b, c; %>

<%! Circle a = new Circle(2.0); %>
```

JSP expression

- A JSP expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.
- Syntax is :

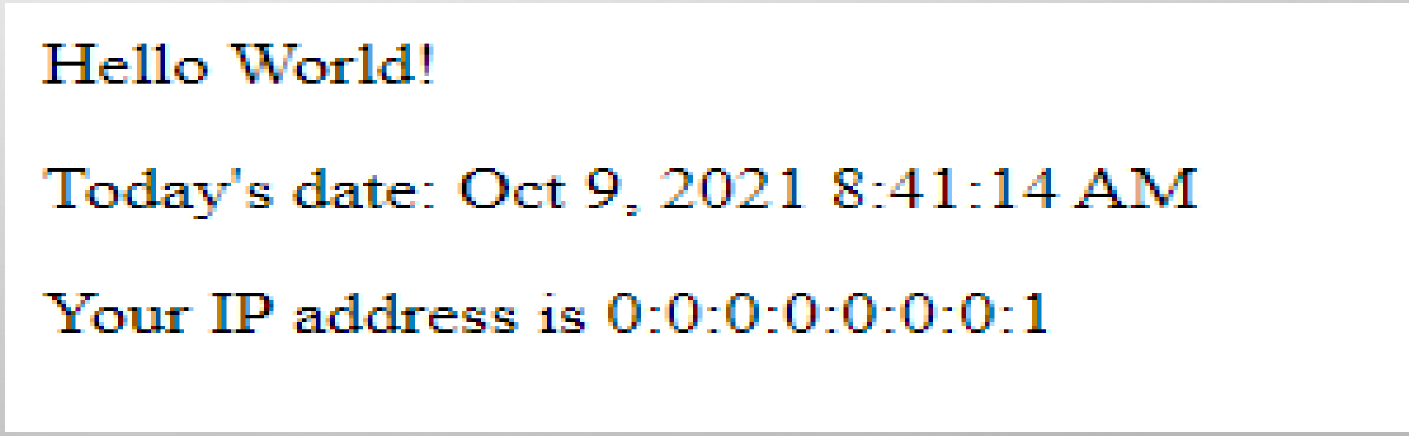
```
<%= expression %>
```

- Because the value of an expression is converted to a String, you can use an expression within a line of text.

```
<p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>
```

- Ex:

```
<html>
  <head><title>Hello World</title></head>
  <body>
    Hello World!<br/>
 <p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>
    <%
      out.println("Your IP address is " + request.getRemoteAddr());
    %>
  </body>
</html>
```



```
Hello World!

Today's date: Oct 9, 2021 8:41:14 AM

Your IP address is 0:0:0:0:0:0:0:1
```

JSP comments

- JSP comment marks text or statements that the JSP container should ignore. A JSP comment is useful when you want to hide or "comment out", a part of your JSP page.

- Following is the syntax of the JSP comments –

```
<%-- This is JSP comment --%>
```

- Ex:

```
<%-- This comment will not be visible in the page source --%>
```

JSP Directives

- A JSP directive affects the overall structure of the servlet class.
- It usually has the following form –

  <%@ directive attribute="value" %>

- There are three types of directive tag –
- <%@ page ... %>

  Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.

- <%@ include ... %>

  Includes a file during the translation phase.

- <%@ taglib ... %>

Declares a tag library, containing custom actions, used in the page

JSP Actions

- You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.
- There is only one syntax for the Action element, as it conforms to the XML standard –

```
<jsp:action_name  attribute = "value" />
```

- Action elements are basically predefined functions:

Ex:

<jsp:include> action

- There are many JSP actions exist ,out of that one is jsp:include

Includes a file at the time the page is requested.

- The syntax looks like this –

```
<jsp:include page = "relative URL" flush = "true" />
```

Path:-The relative URL of the page to be included.

Flush:-
The boolean attribute determines whether the included resource has its buffer flushed before it is included.

JSP Actions

Let us define the following two files (a)date.jsp and (b) main.jsp as follows –

Following is the content of the date.jsp file –

```
<p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>
```

Following is the content of the main.jsp file –

```
<html> <head>
    <title>The include Action Example</title></head>
  <body>  <center>
      <h2>The include action Example</h2>
      <jsp:include page = "date.jsp" flush = "true" />
    </center> </body>
</html>
```

JSP implicit objects

- JSP supports nine automatically defined variables, which are also called implicit objects. Out of them 3 variables are −

1) request

This is the HttpServletRequest object associated with the request.

2) response

This is the HttpServletResponse object associated with the response to the client.

3) Out

This is the PrintWriter object used to send output to the client.

JSTL library

- The Java Server Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates the core functionality common to many JSP applications.

- JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags.
- It also provides a framework for integrating the existing custom tags with the JSTL tags.
- To install JSTL library compatible with our servlet and JSP version

Link is:

http://www.java2s.com/Code/Jar/j/Downloadjavaxservletjspjstl30jar.htm

- To use any of the libraries, you must include a <taglib> directive at the top of each JSP that uses the library.

JSTL library

Classification of The JSTL Tags:

The JSTL tags can be classified, according to their functions, into the following JSTL tag library groups that can be used when creating a JSP page −

- Core Tags
- Formatting tags
- SQL tags
- XML tags
- JSTL Functions

Note :

That all the JSTL standard tags URL starts with https://java.sun.com/jsp/jstl/ and we can use any prefix we want but it’s best practice to use the prefix defined above

JSTL library

Classification of The JSTL Tags-

1) Core Tags:

JSTL Core tags provide support for

1. iteration,
2. conditional logic,
3. catch exception,
4. url,
5. forward or redirect response etc.

To use JSTL core tags, we should include it in the JSP page like below.

```
<%@ taglib uri="https://java.sun.com/jsp/jstl/core"  prefix="c" %>
```

JSTL library

Classification of The JSTL Tags-

2) Formatting Tags:

JSTL Formatting tags are provided for

1. formatting of Numbers,
2. Dates and i18n support through locales and resource bundles.

We can include these jstl tags in JSP with below syntax:

```
<%@ taglib uri="https://java.sun.com/jsp/jstl/fmt"  prefix="fmt"  %>
```

JSTL library

Classification of The JSTL Tags-

3) JSTL SQL Tags:

1. JSTL SQL Tags provide support for interaction with relational databases such as Oracle, MySql etc.
2. Using JSTL SQL tags we can run database queries,

we include these JSTL tags in JSP with below syntax:

```
<%@ taglib uri="https://java.sun.com/jsp/jstl/sql"  prefix="sql"  %>
```

JSTL library

Classification of The JSTL Tags-

4) JSTL XML Tags:

1. JSTL XML tags are used to work with XML documents such as parsing XML, transforming XML data and XPath expressions evaluation.
2. Syntax to include JSTL XML tags in JSP page is:

```
<%@ taglib uri="https://java.sun.com/jsp/jstl/xml"  prefix="x" %>
```

JSTL library

Classification of The JSTL Tags-

5) JSTL Functions Tags:

JSTL tags provide several functions that we can use,

1. to perform common operation,
2. most of them are for String manipulation such as String Concatenation, Split String etc.

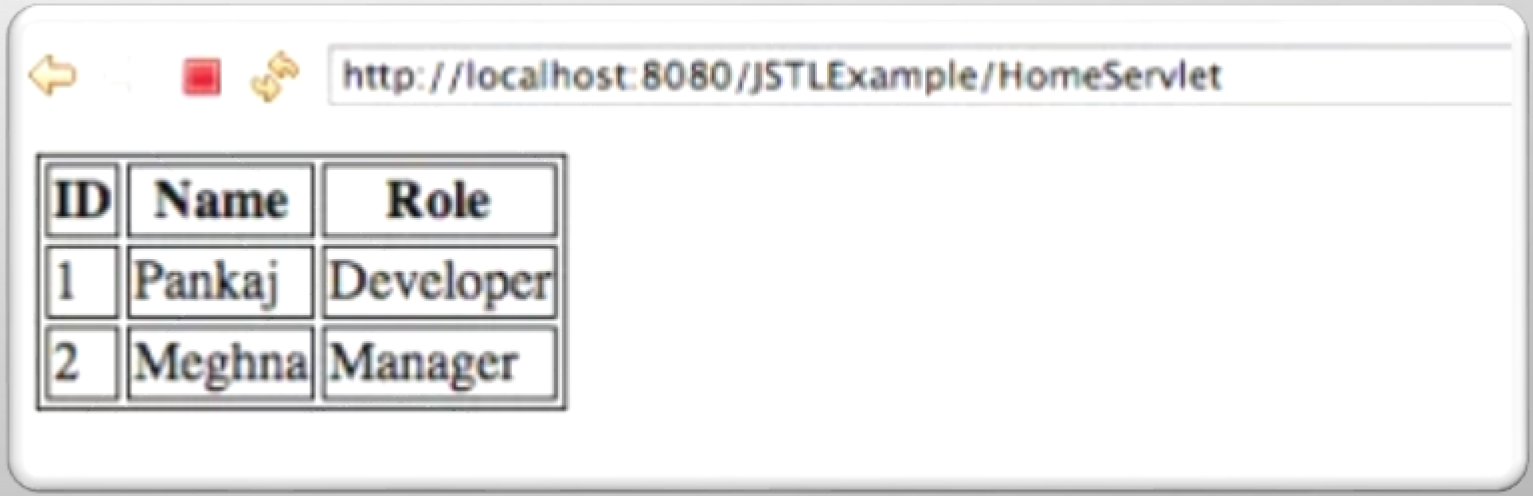
- Syntax to include JSTL functions in JSP page is:

```
<%@ taglib uri="https://java.sun.com/jsp/jstl/functions"  prefix="fn" %>
```

EXAMPLE OF JSTL LIBRARY

- One Example of core tag :

```
<c:foreach items="${requestscope.emplist}"  var="emp">

<tr><td><c:out value="${emp.id}"></c:out></td>

<td><c:out value="${emp.name}"></c:out></td>

<td><c:out value="${emp.role}"></c:out></td></tr>

</c:foreach>
```



http://localhost:8080/JSTLExample/HomeServlet

ID	Name	Role
1	Pankaj	Developer
2	Meghna	Manager

CREATING HTML FORMS WITH JAVASERVER PAGES

- Using GET Method-

The GET method sends the encoded user information appended to the page request.

The page and the encoded information are separated by the ? character as follows −

http://www.test.com/hello?key1=value1&key2=value2

The GET method is the default method to pass information from the browser to the web server and it produces a long string that appears in your browser's Location:box.

- Using GET Method-

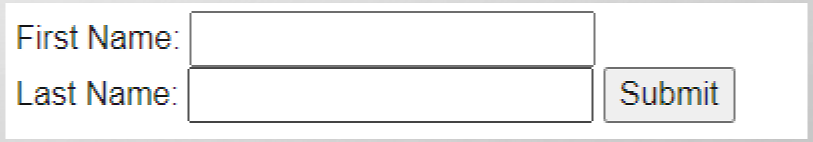
Ex:

```
<html>
   <head>
      <title>Using GET Method to Read Form Data</title>
   </head>

   <body>
      <h1>Using GET Method to Read Form Data</h1>
      <ul>
         <li><p><b>First Name:</b>
            <%= request.getParameter("first_name")%>
         </p></li>
         <li><p><b>Last  Name:</b>
            <%= request.getParameter("last_name")%>
         </p></li>
      </ul>

   </body>
</html>
```

```
<html>
   <body>

      <form action = "main.jsp" method = "GET">
         First Name: <input type = "text" name = "first_name">
         <br />
         Last Name: <input type = "text" name = "last_name" />
         <input type = "submit" value = "Submit" />
      </form>

   </body>
</html>
```

Run the program as :

http://localhost:8080/Hello.html



- Using POST method-

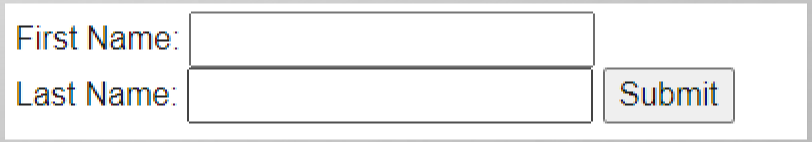
Ex:

```
<html>
   <head>
      <title>Using GET Method to Read Form Data</title>
   </head>

   <body>
      <h1>Using GET Method to Read Form Data</h1>
      <ul>
         <li><p><b>First Name:</b>
            <%= request.getParameter("first_name")%>
         </p></li>
         <li><p><b>Last  Name:</b>
            <%= request.getParameter("last_name")%>
         </p></li>
      </ul>

   </body>
</html>
```

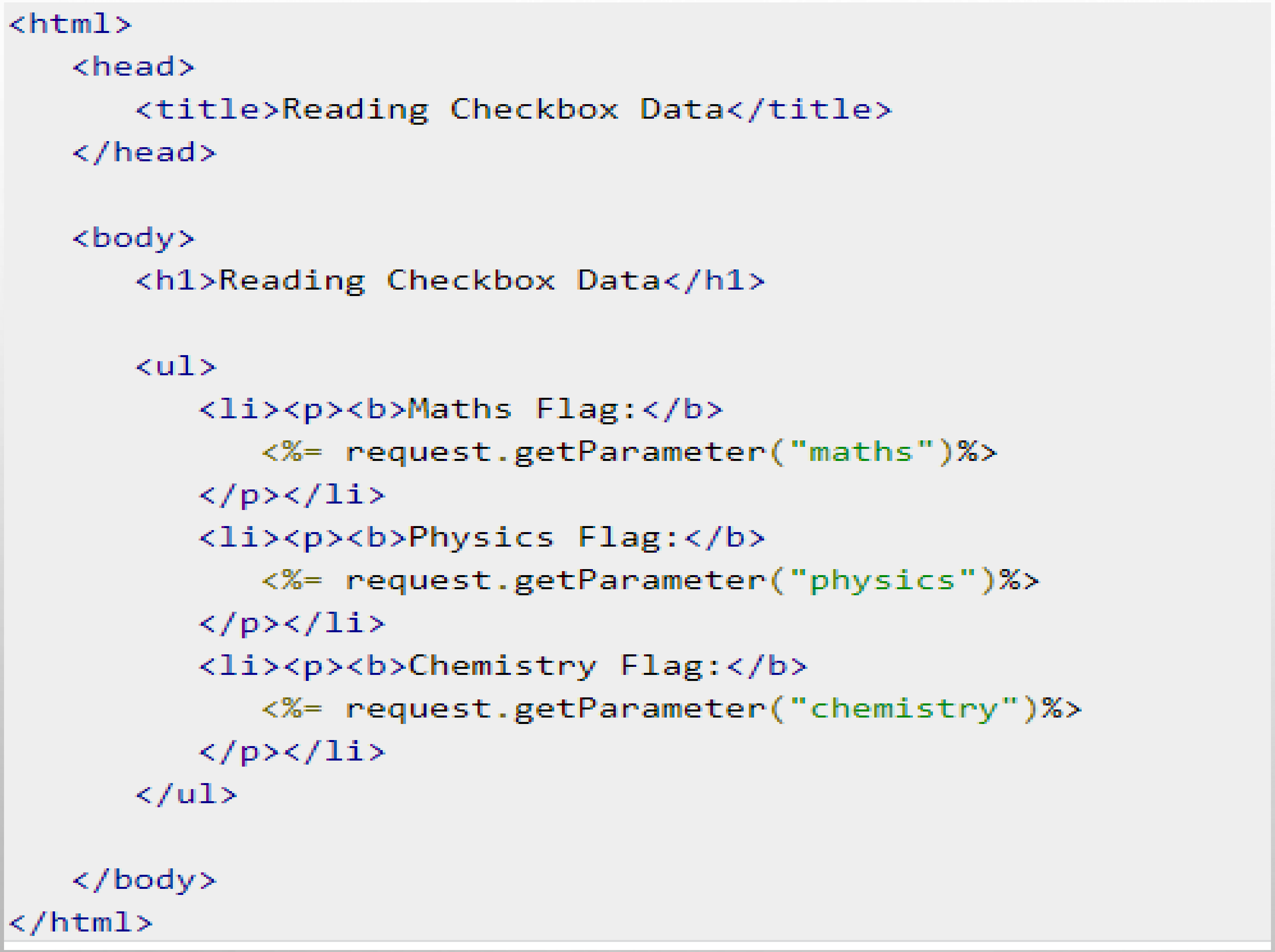
```
<html>
   <body>

      <form action = "main.jsp" method = "POST">
         First Name: <input type = "text" name = "first_name">
         <br />
         Last Name: <input type = "text" name = "last_name" />
         <input type = "submit" value = "Submit" />
      </form>


   </body>
</html>
```

Run the program as :

http://localhost:8080/Hello.html



- Passing checkbox data to JSP program:

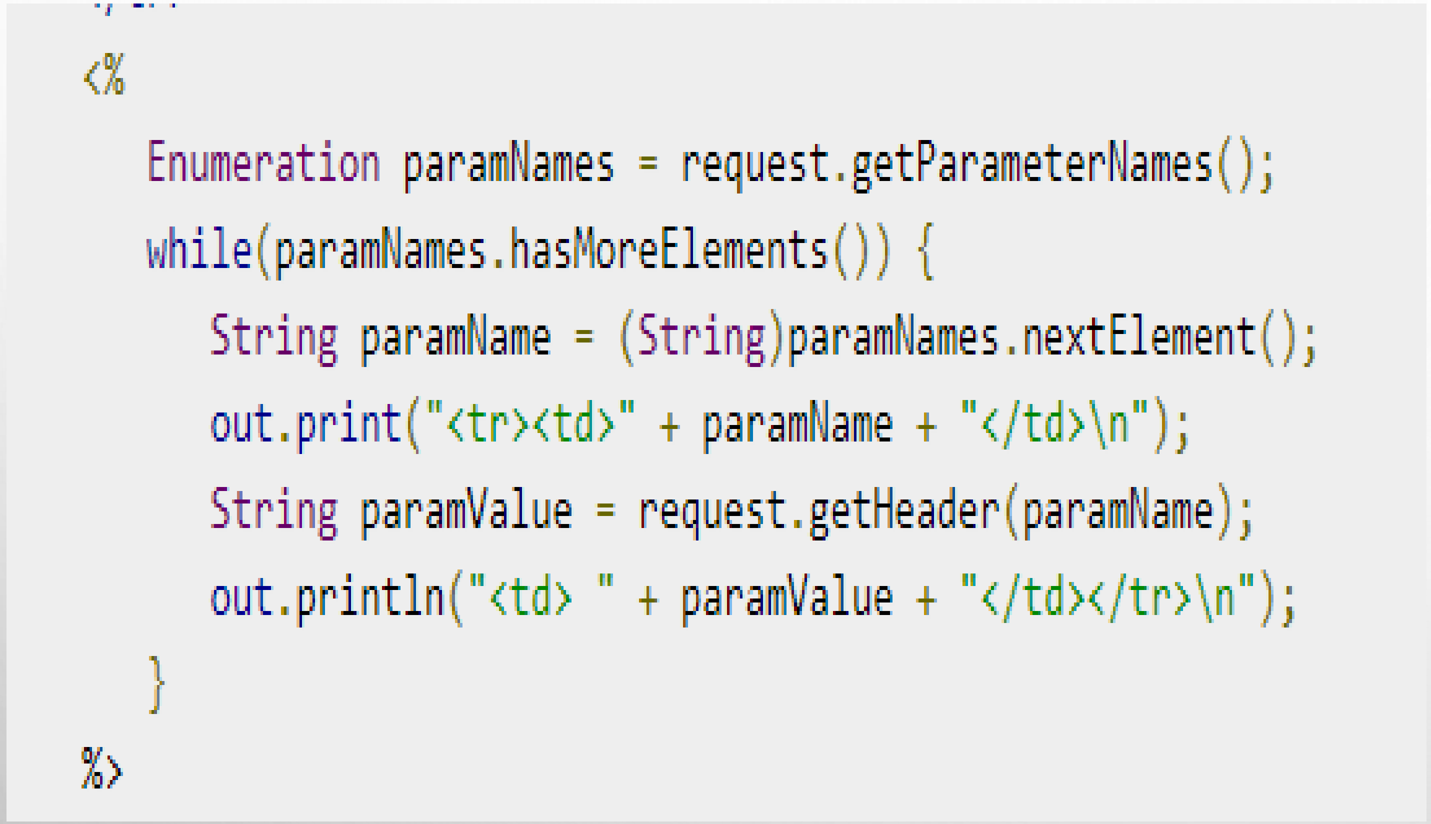


```
<html>
   <head>
      <title>Reading Checkbox Data</title>
   </head>

   <body>
      <h1>Reading Checkbox Data</h1>

      <ul>
         <li><p><b>Maths Flag:</b>
            <%= request.getParameter("maths")%>
         </p></li>
         <li><p><b>Physics Flag:</b>
            <%= request.getParameter("physics")%>
         </p></li>
         <li><p><b>Chemistry Flag:</b>
            <%= request.getParameter("chemistry")%>
         </p></li>
      </ul>

   </body>
</html>
```

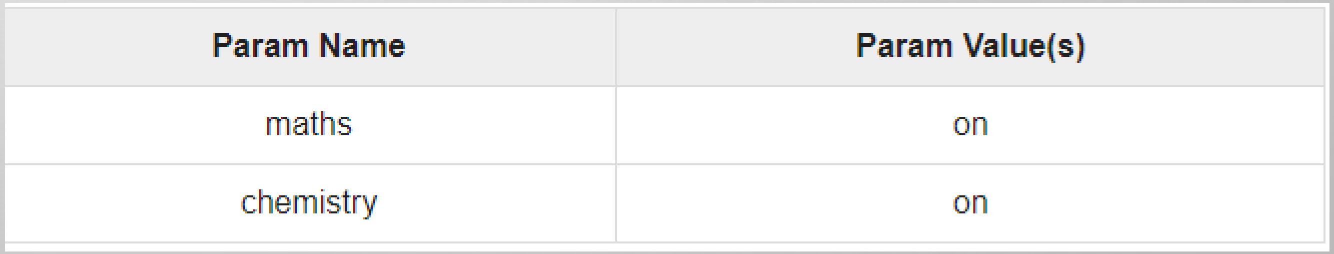
- Reading all form parameters:
- Following is a generic example which uses getParameterNames() method of HttpServletRequest to read all the available form parameters.
- This method returns an Enumeration that contains the parameter names in an unspecified order.

- Once we have an Enumeration, we can loop down the Enumeration in the standard manner, using the hasMoreElements() method to determine when to stop and using the nextElement() method to get each parameter name.

Ex:



```
<%
   Enumeration paramNames = request.getParameterNames();
   while(paramNames.hasMoreElements()) {
      String paramName = (String)paramNames.nextElement();
      out.print("<tr><td>" + paramName + "</td>\n");
      String paramValue = request.getHeader(paramName);
      out.println("<td> " + paramValue + "</td></tr>\n");
   }
%>
```



Param Name	Param Value(s)
maths	on
chemistry	on