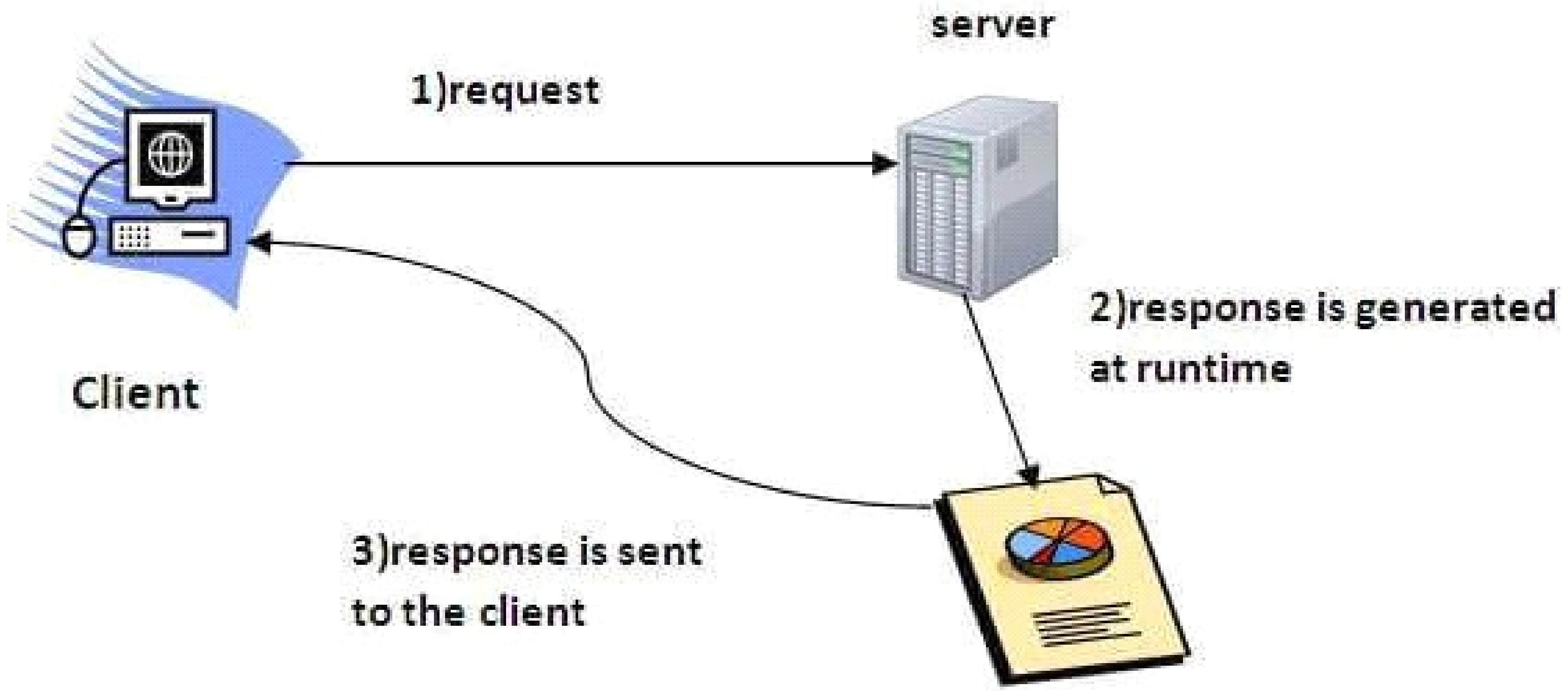
SERVLET

1) Servlet technology is used to create a web application (resides at server side and generates a dynamic web page)..
2) Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
3) Servlet is a web component that is deployed on the server to create a dynamic web page.


server
1)request
Client
2)response is generated
at runtime
3)response is sent
to the client

Execution of Servlets basically involves six basic steps:

(1) The clients send the request to the webserver.

(2) The web server receives the request.

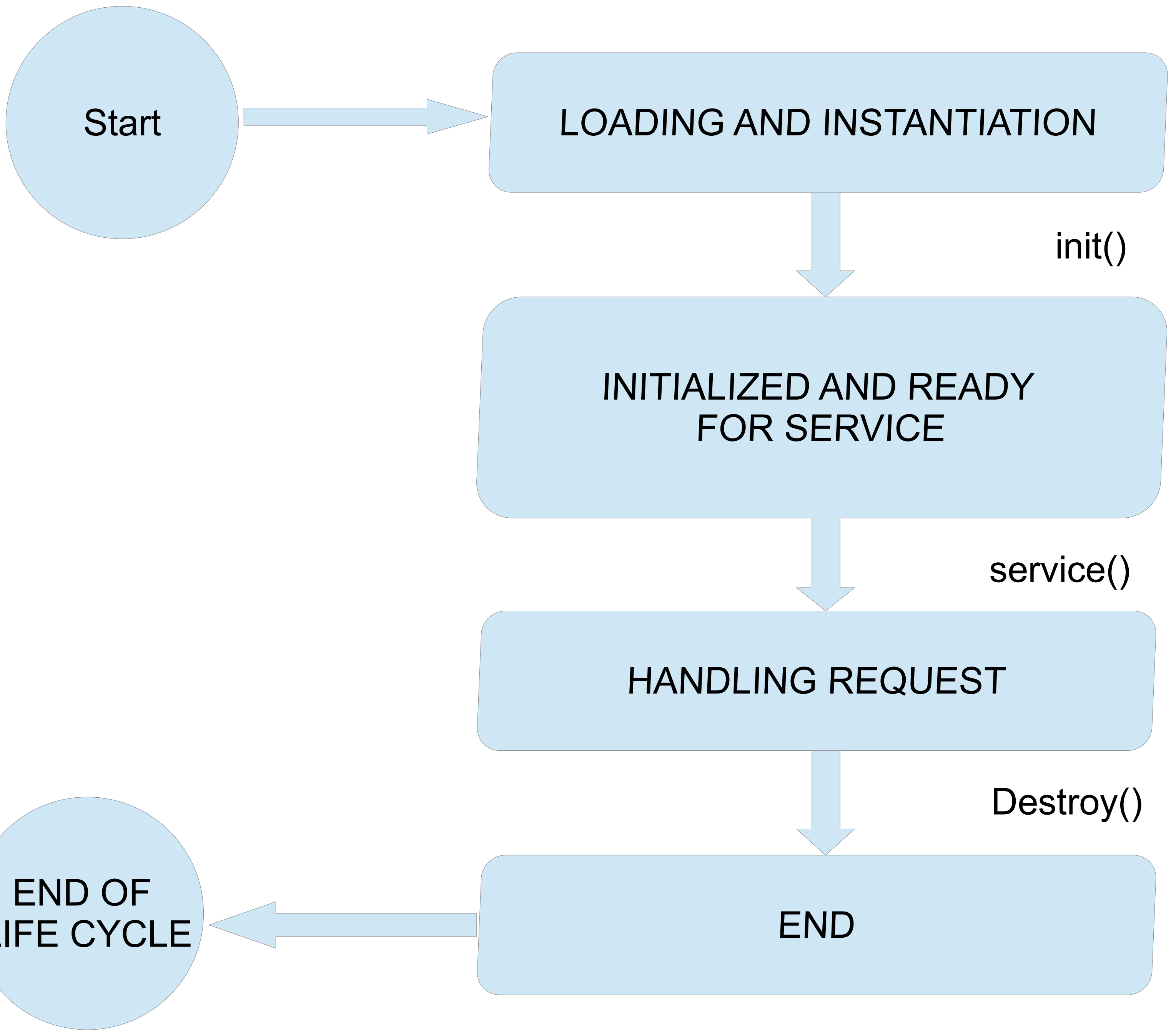
(3) The web server passes the request to the corresponding servlet.

(4) The servlet processes the request and generates the response in the form of output.

(5) The servlet sends the response back to the webserver.

(6) The web server sends the response back to the client and the client browser displays it on the screen.

Servlet life cycle



public void init ():

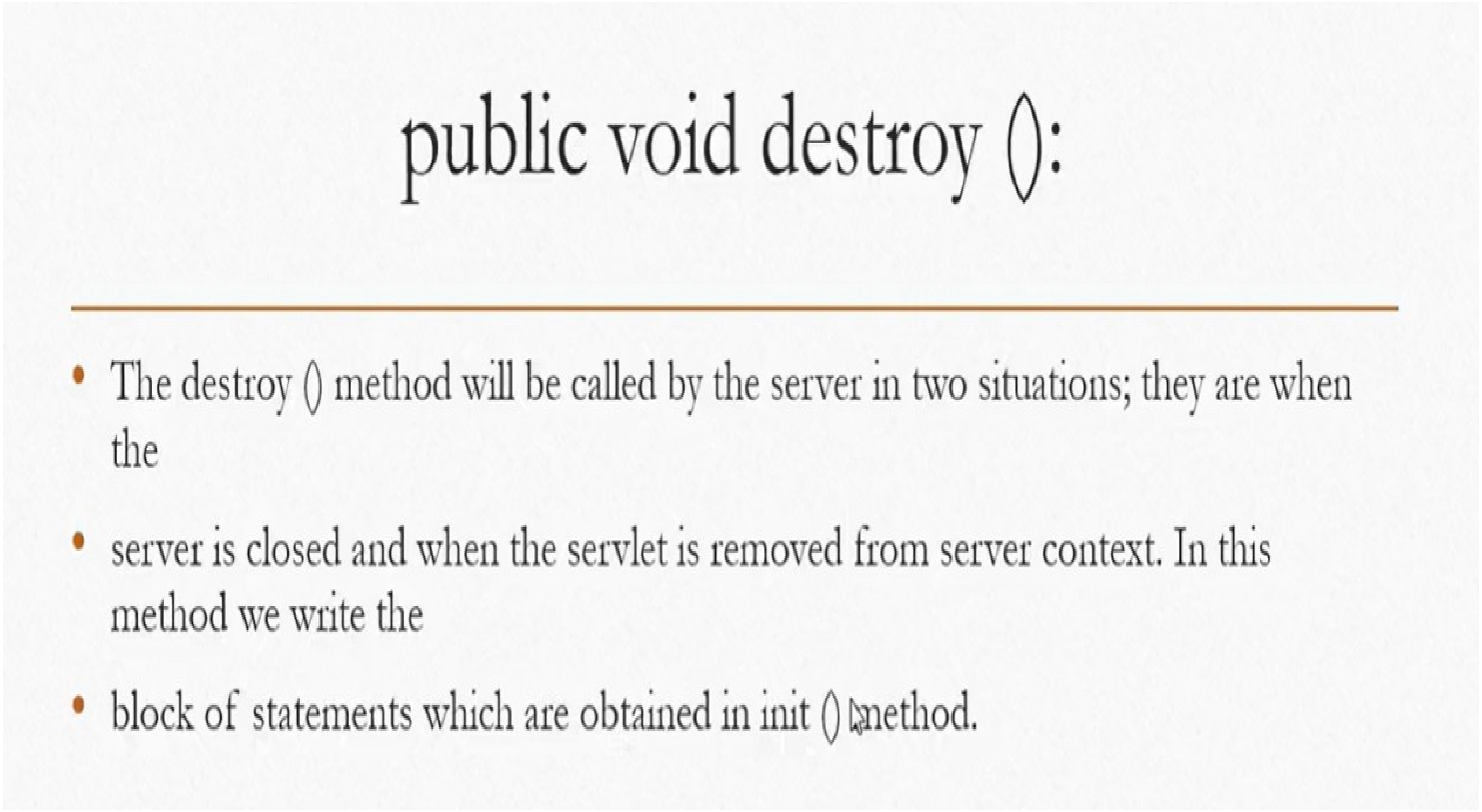
- Whenever client makes a request to a servlet, the server will receive the request and it automatically calls init () method i.e., init () method will be called only one time by the server whenever we make first request.
- In this method, we write the block of statements which will perform one time operations, such as, opening the file, database connection, initialization of parameters, etc.

public void service (ServletRequest, ServletResponse):

- After calling init () method, service () method will be called when we make first request from second request to further subsequent requests, server will call only service method. Therefore, service () method will be called each and every time as and when we make a request.
- In service () method we write the block of statements which will perform repeated operations such as retrieving data from database, retrieving data from file, modifications of parameters data, etc. Hence, in service () method we always write business logic.

public void service (ServletRequest, ServletResponse):

- Whenever control comes to service () method the server will create two objects of ServletRequest and ServletResponse interfaces. Object of ServletRequest contains the data which is passed by client. After processing client data, the resultant data must be kept in an object of ServletResponse.
- An object of ServletRequest and ServletResponse must be used always within the scope of service () method only i.e., we cannot use in init () method and destroy () method.
- Once the service () method is completed an object of ServletRequest and an object of ServletResponse will be destroyed.



public void destroy ():

- The destroy () method will be called by the server in two situations; they are when the
- server is closed and when the servlet is removed from server context. In this method we write the
- block of statements which are obtained in init () method.

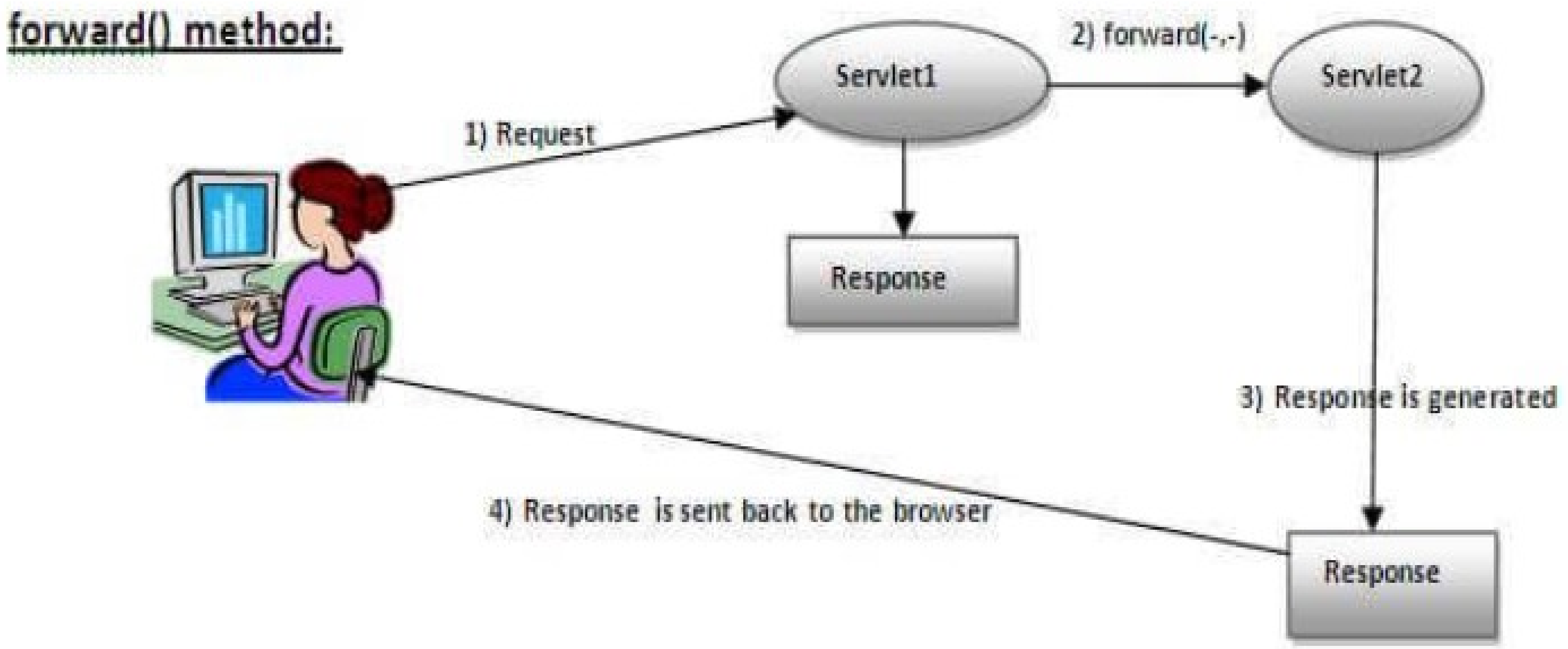
RequestDispatcher in Servlet

- The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also.
- Two methods:

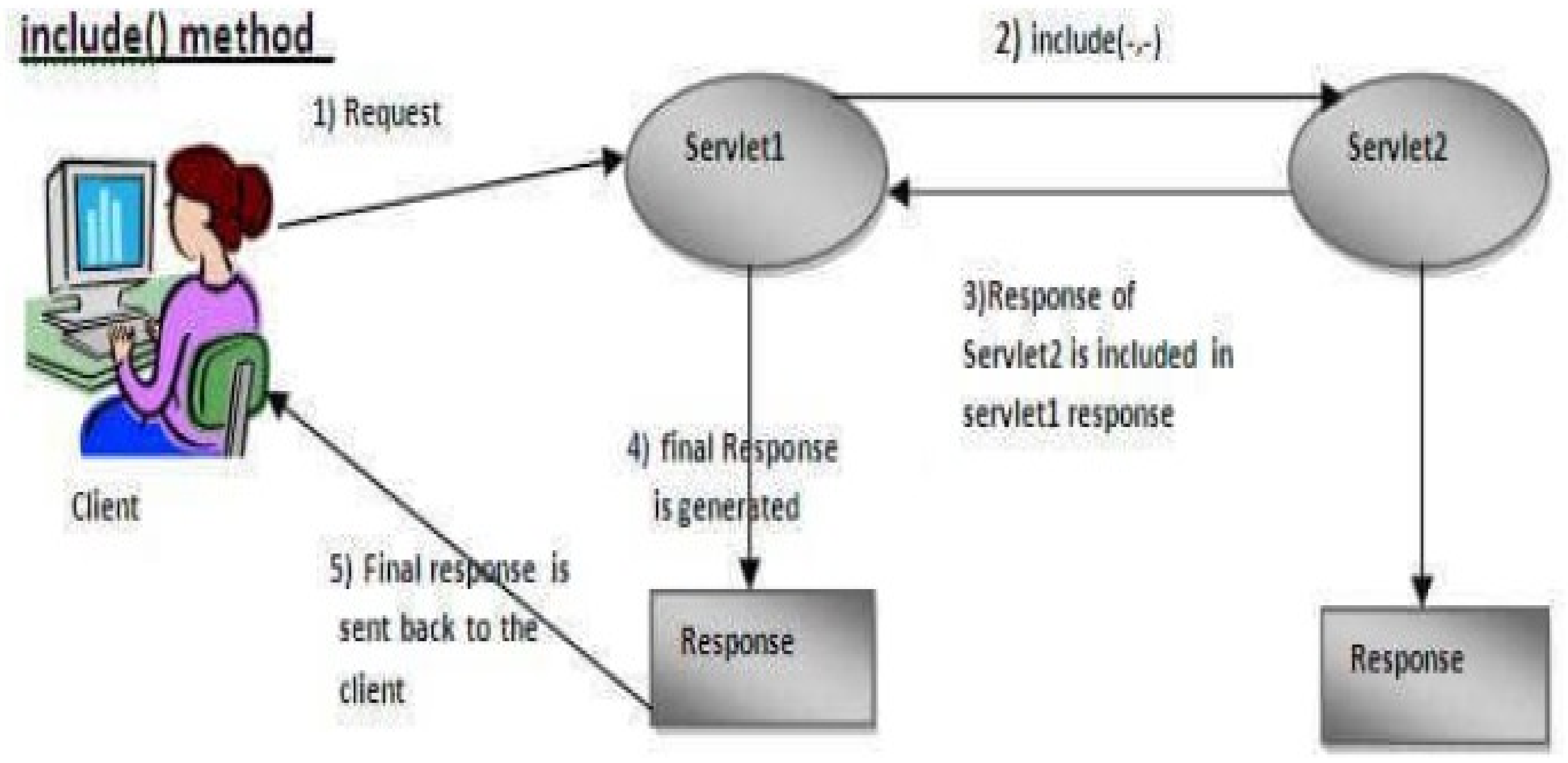
  (1) forward()

  (2) include()

Forward Method



Include Method


include() method
1) Request
2) include(-,-)
Servlet1
Servlet2
3)Response of Servlet2 is included in servlet1 response
4) final Response is generated
Client
5) Final response is sent back to the client
Response
Response

SendRedirect in servlet

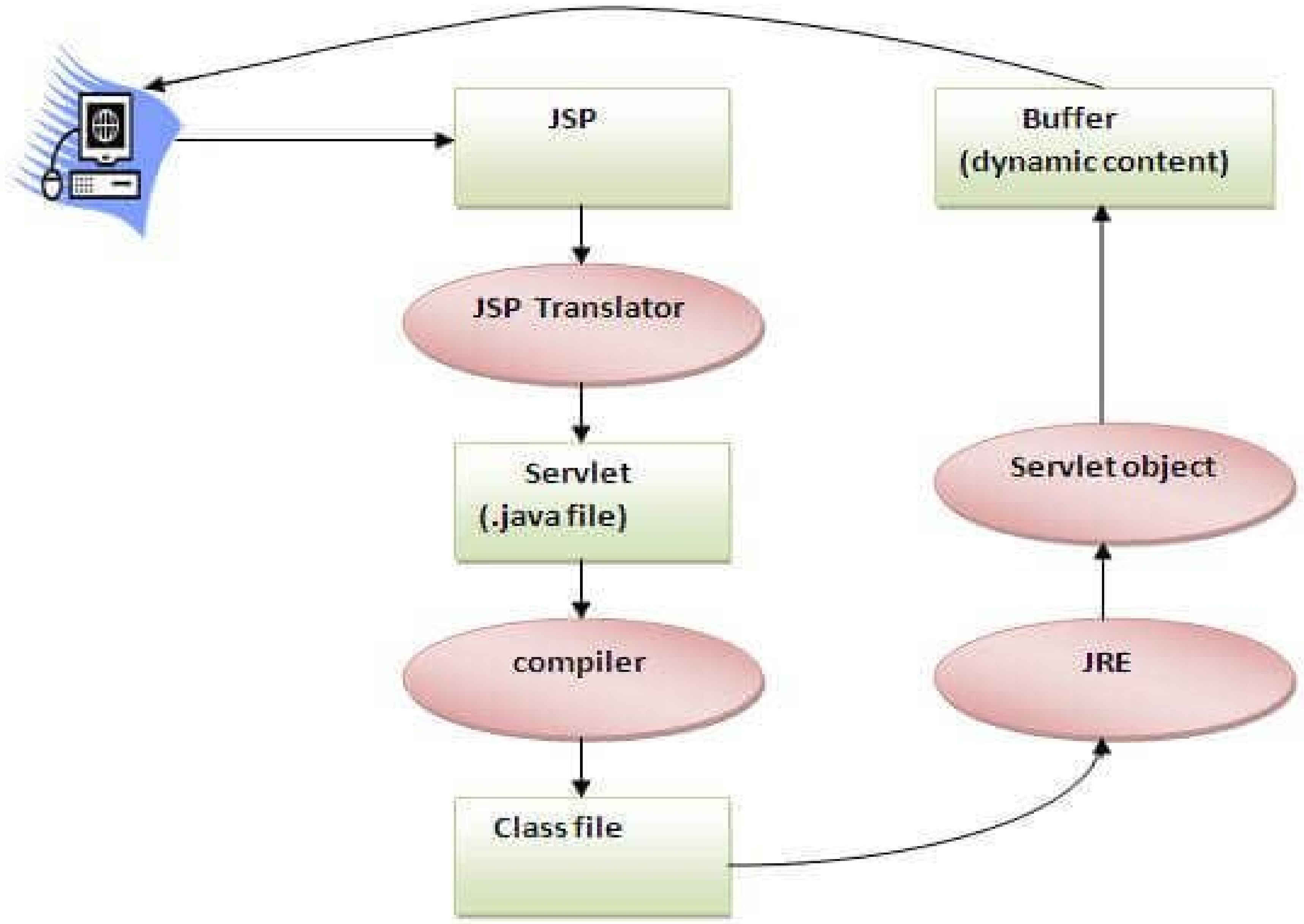
- The sendRedirect() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, jsp or html file.

- It accepts relative as well as absolute URL.

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.
- JSP is similar to HTML pages, but they also contain Java code executed on the server side.
- A JSP page is a file with a ".jsp" extension that can contain a combination of HTML Tags and JSP codes.

The Lifecycle of a JSP Page



JSP Scripting elements

- The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

1) scriptlet tag
2) expression tag
3) declaration tag

(1) scriptlet tag

```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>
</html>
```

(2) expression tag

```
<html>
<body>
<%= "welcome to jsp" %>
</body>
</html>
```

(3) declaration tag

```
<html>

<body>

<%! int data=50; %>

<%= "Value of the variable is:"+data %>

</body>

</html>
```

JSP Implicit Objects

(1) Request implicit object

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

welcome.jsp

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

(2)response implicit object

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

welcome.jsp

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

(3)Session implicit object

-
  ```
  <html>
  <body>
  <form action="welcome.jsp">
  <input type="text" name="uname">
  <input type="submit" value="go"><br/>
  </form>
  </body>
  </html>
  ```

welcome.jsp

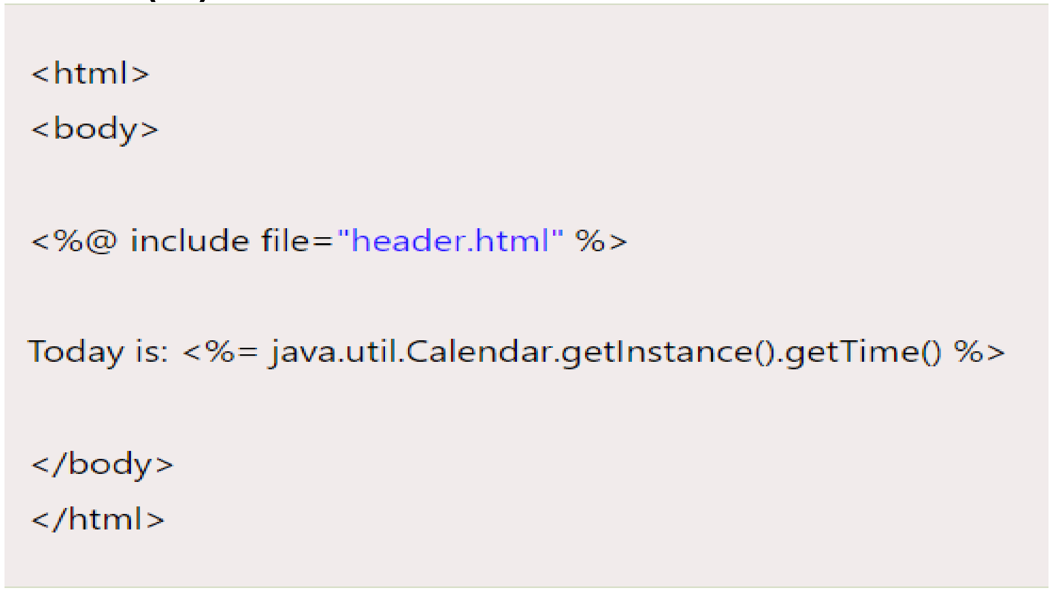
```
<html>
<body>
<%

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

```
<html>
<body>
<%

String name=(String)session.getAttribute("user");
out.print("Hello "+name);

%>
</body>
</html>
```

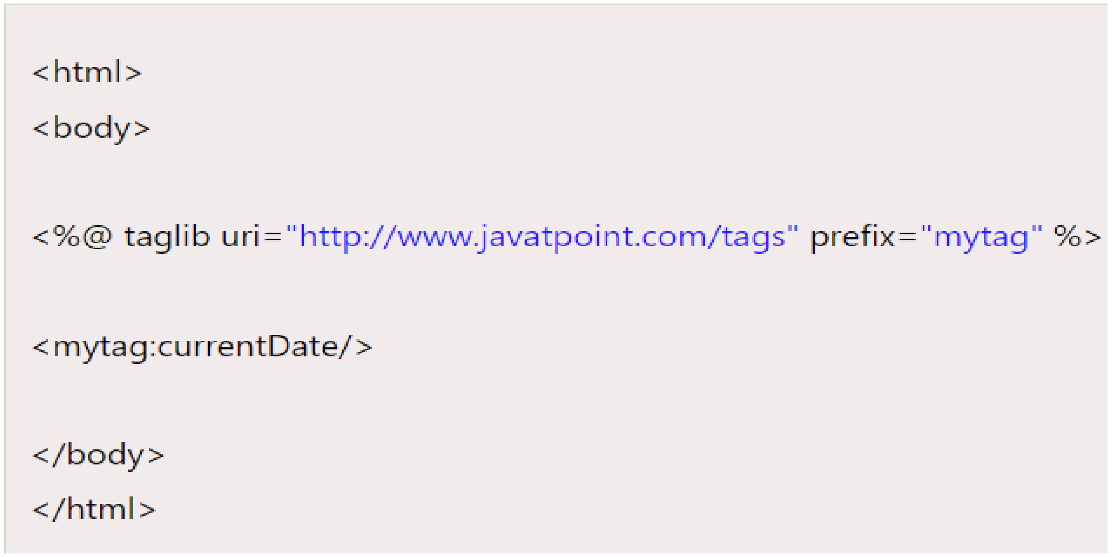
JSP directives(The jsp directives are messages that tells the web container how to translate a JSP page into the corresponding servlet.)

(1) Include Directive



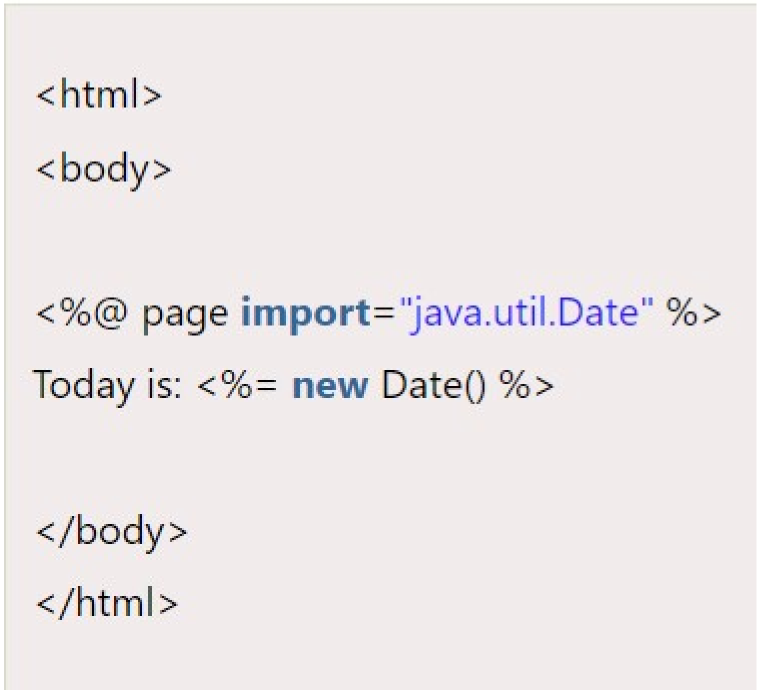
```
<html>
<body>

<%@ include file="header.html" %>

Today is: <%= java.util.Calendar.getInstance().getTime() %>

</body>
</html>
```

(2) Taglib directive



```
<html>
<body>

<%@ taglib uri="http://www.javatpoint.com/tags" prefix="mytag" %>

<mytag:currentDate/>

</body>
</html>
```

(3)Page directive

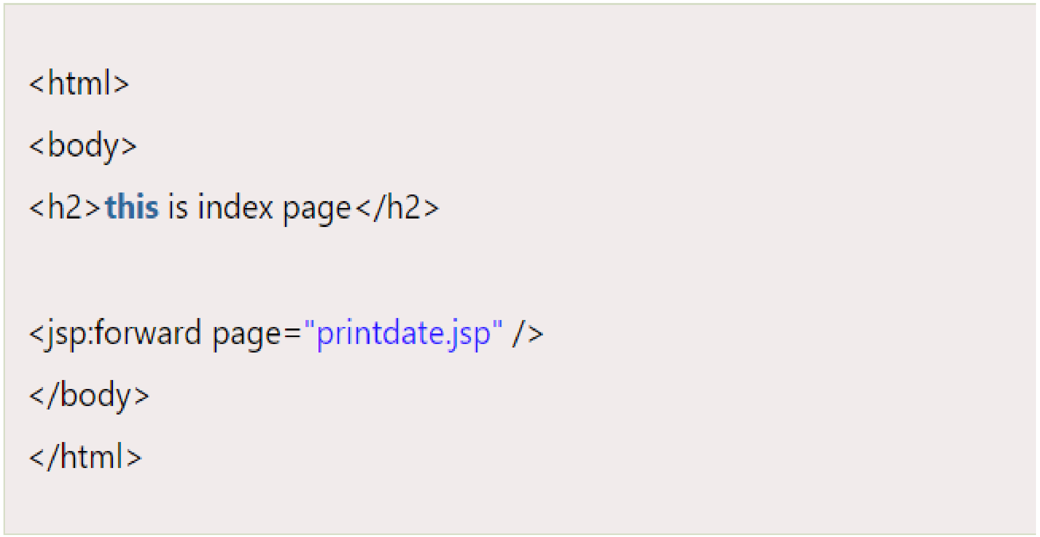


```
<html>
<body>

<%@ page import="java.util.Date" %>
Today is: <%= new Date() %>

</body>
</html>
```

JSP Action Tags(There are many JSP action tags or elements. Each JSP action tag is used to perform some specific tasks)

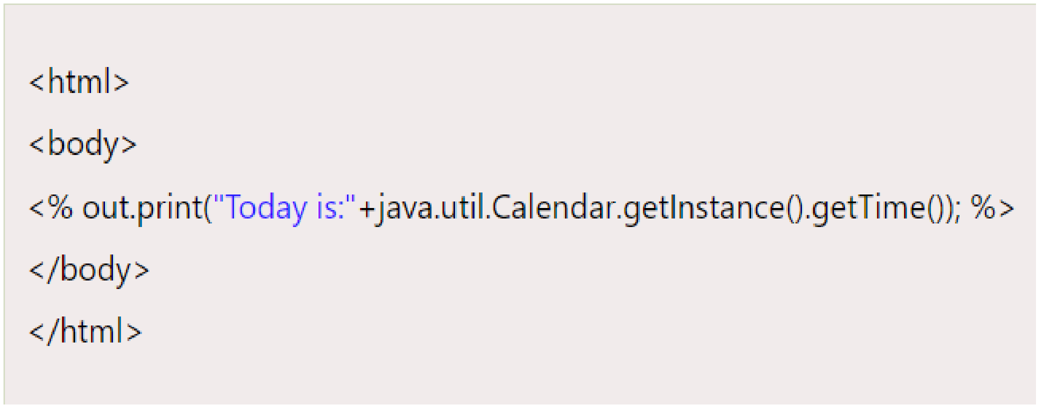
Forward Action

index.jsp



```
<html>
<body>
<h2>this is index page</h2>

<jsp:forward page="printdate.jsp" />
</body>
</html>
```

printdate.jsp

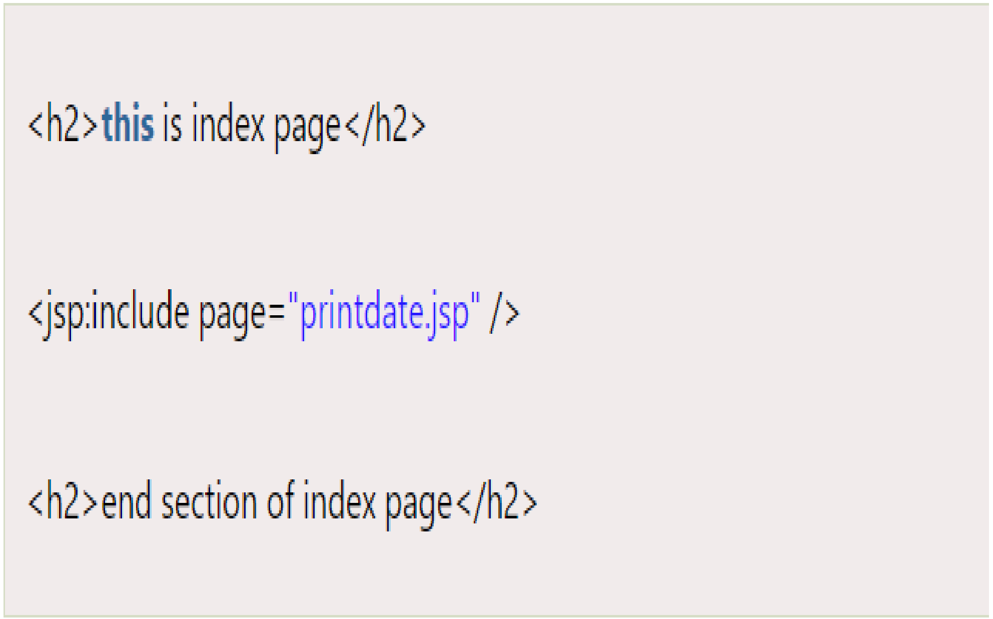


```
<html>
<body>
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
</body>
</html>
```

Include Action

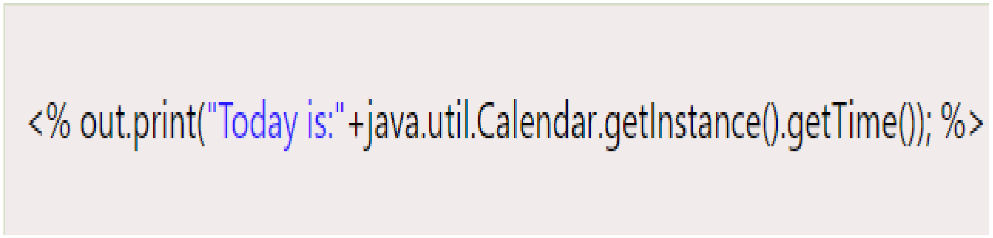
-

File: index.jsp



```
<h2>this is index page</h2>

<jsp:include page="printdate.jsp" />

<h2>end section of index page</h2>
```

File: printdate.jsp

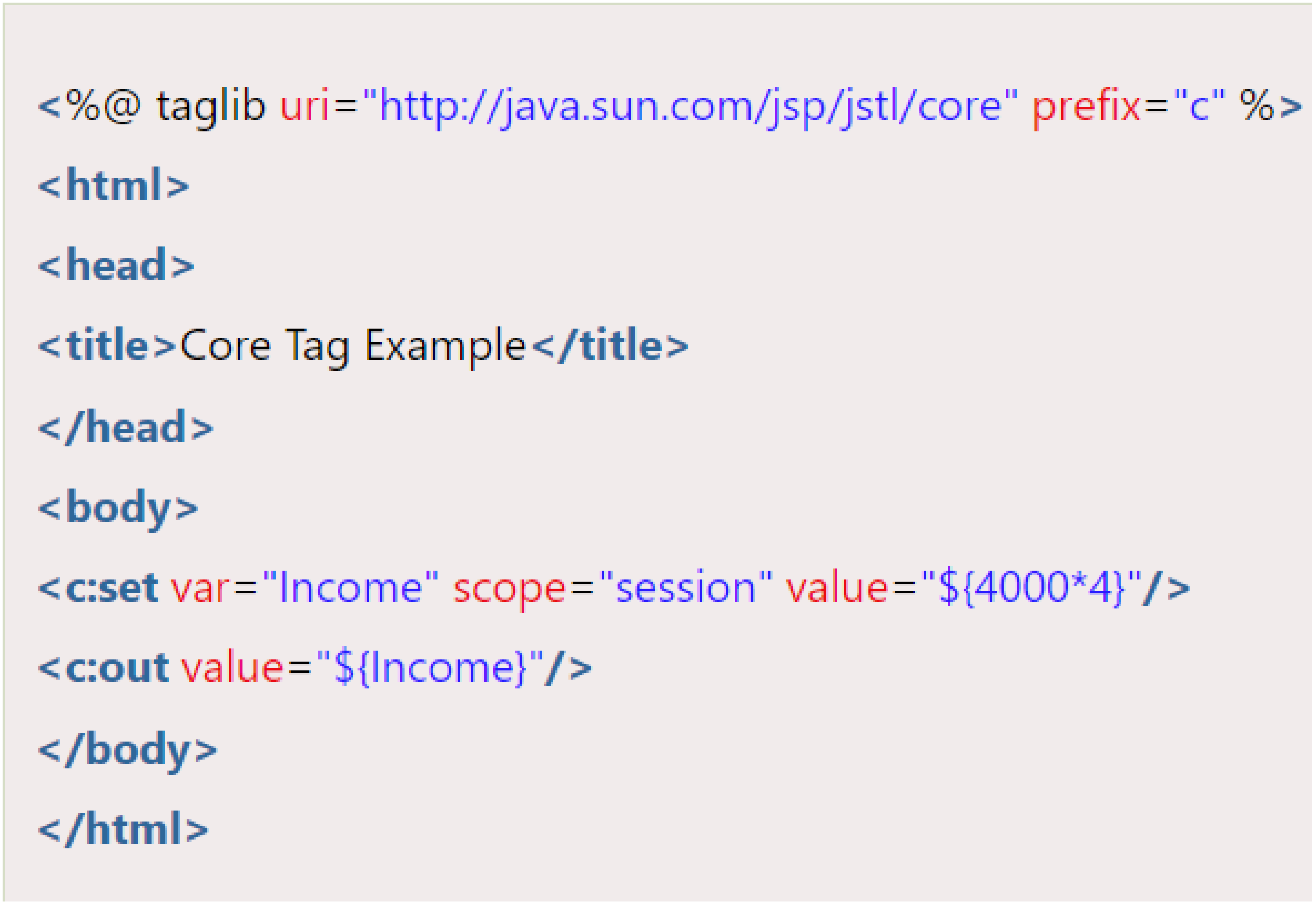


```
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
```

JSTL

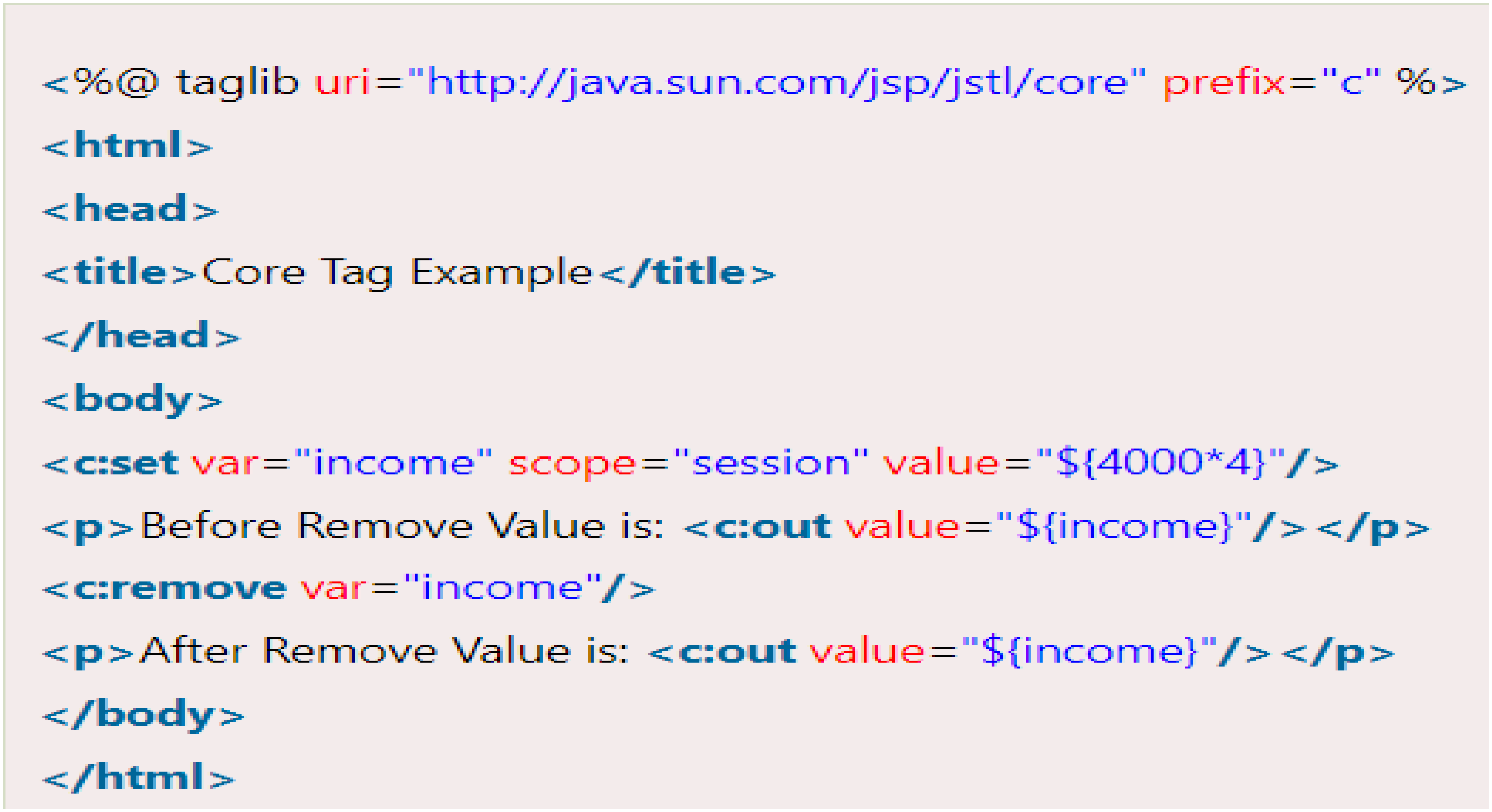
JSTL provides many tags that simplify the JSP,It avoids the use of scriptlet tag.

<c:set> Tag



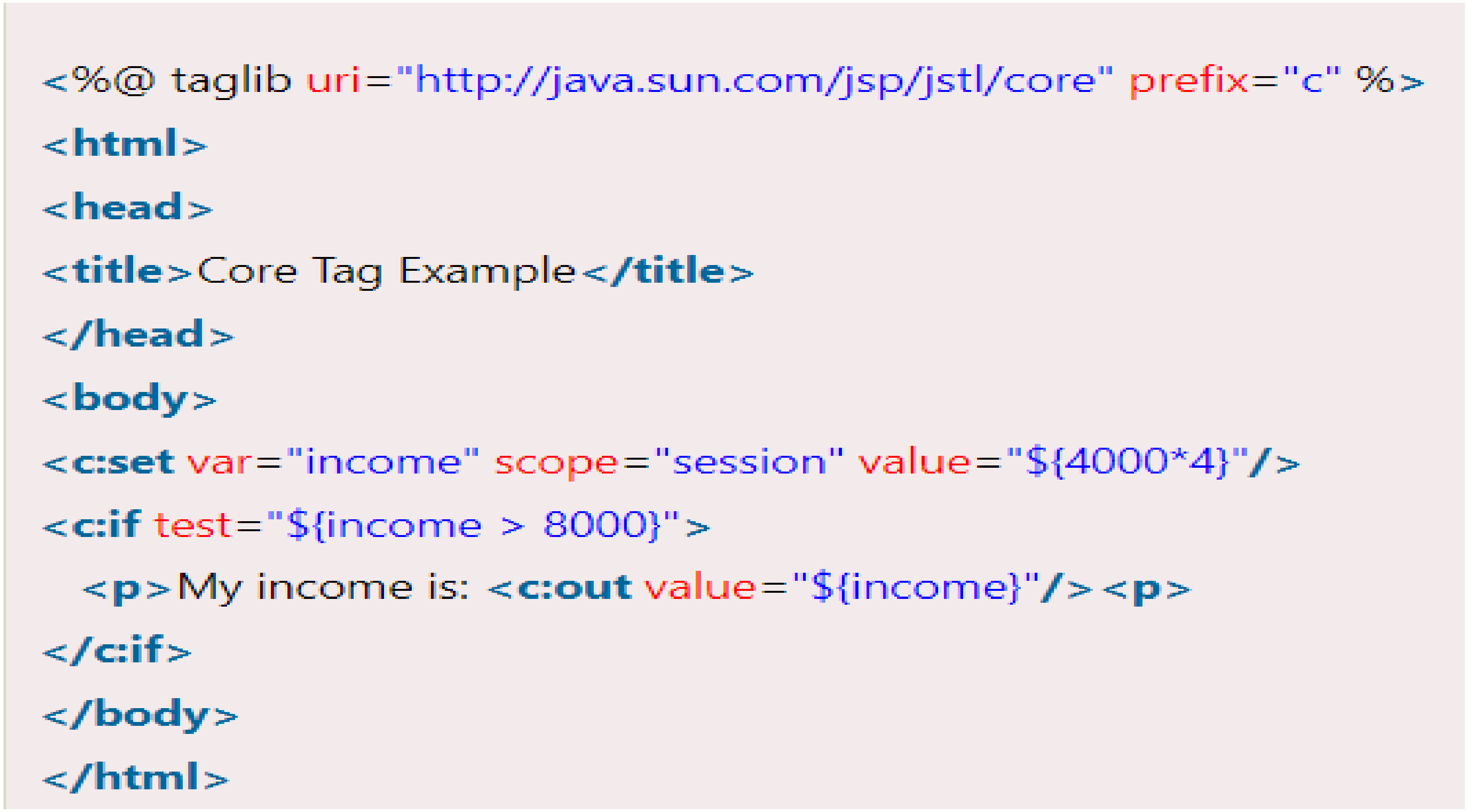
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="Income" scope="session" value="${4000*4}"/>
<c:out value="${Income}"/>
</body>
</html>
```

<c:remove> Tag



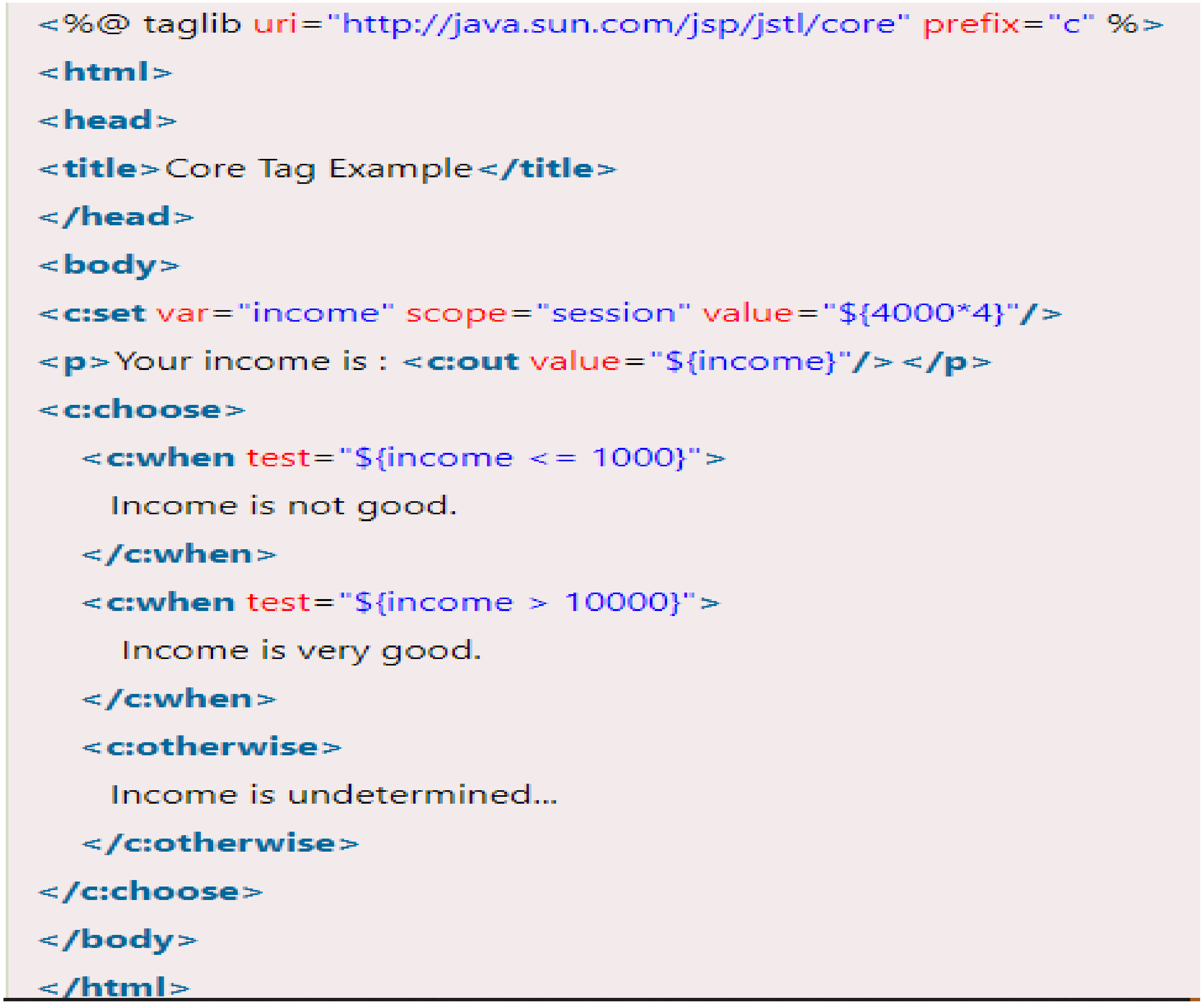
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${4000*4}"/>
<p>Before Remove Value is: <c:out value="${income}"/></p>
<c:remove var="income"/>
<p>After Remove Value is: <c:out value="${income}"/></p>
</body>
</html>
```

<c:if> Tag



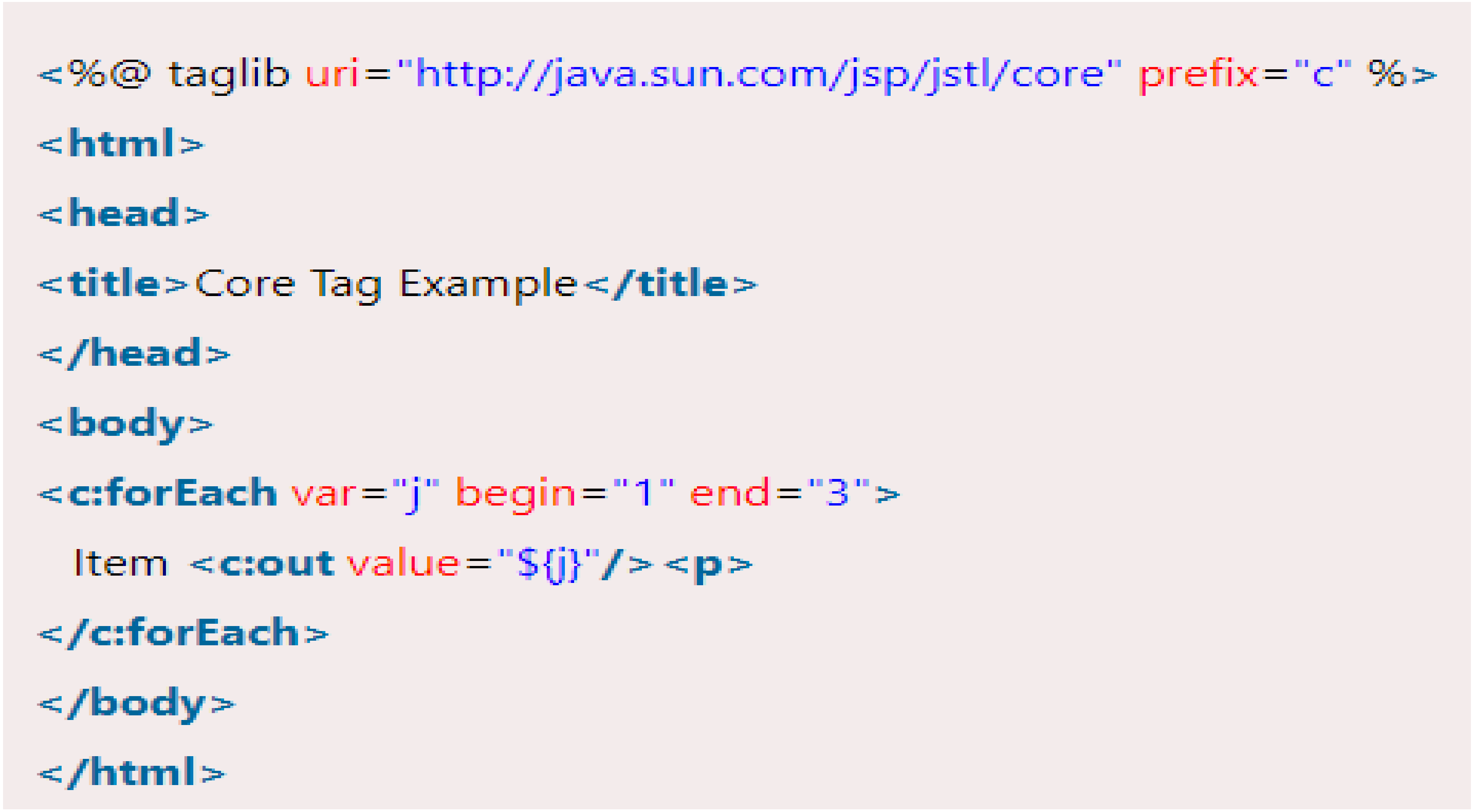
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${4000*4}"/>
<c:if test="${income > 8000}">
   <p>My income is: <c:out value="${income}"/><p>
</c:if>
</body>
</html>
```

<c:choose>, <c:when>, <c:otherwise> tag



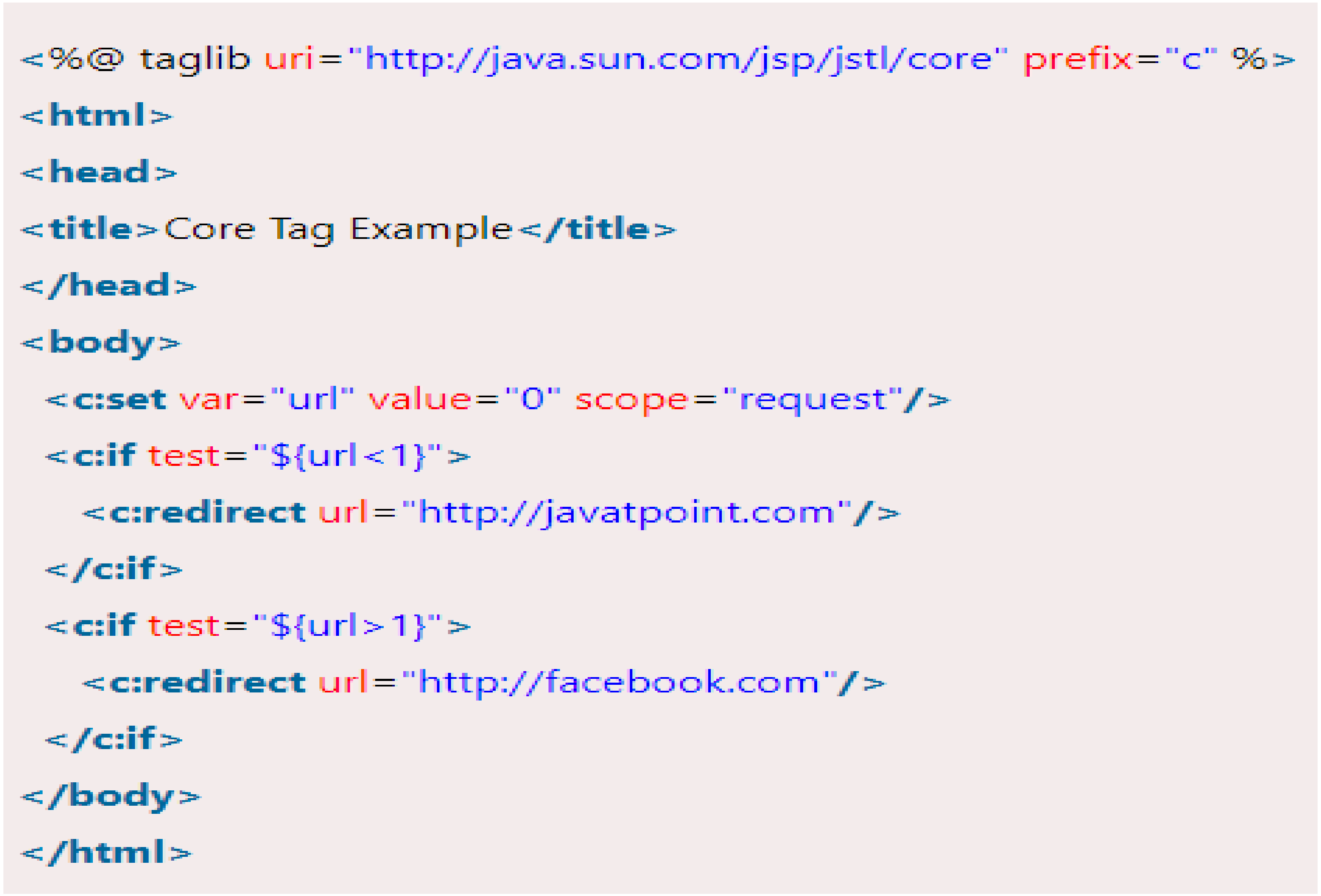
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${4000*4}"/>
<p>Your income is : <c:out value="${income}"/></p>
<c:choose>
    <c:when test="${income <= 1000}">
      Income is not good.
    </c:when>
    <c:when test="${income > 10000}">
       Income is very good.
    </c:when>
    <c:otherwise>
      Income is undetermined...
    </c:otherwise>
</c:choose>
</body>
</html>
```

<c:forEach> Tag



```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:forEach var="j" begin="1" end="3">
  Item  <c:out value="${j}"/><p>
</c:forEach>
</body>
</html>
```

<c:redirect> Tag

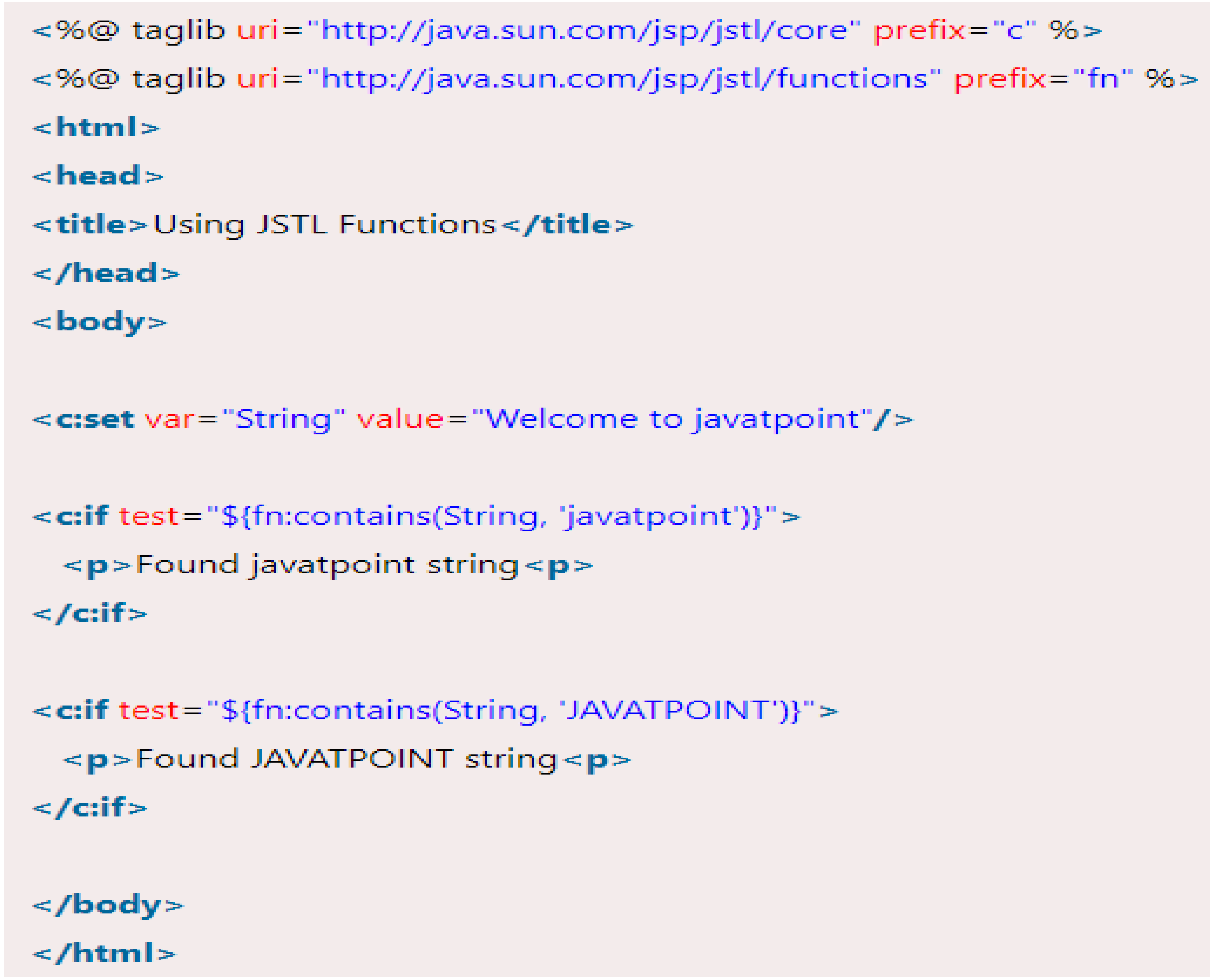


```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
  <c:set var="url" value="0" scope="request"/>
  <c:if test="${url<1}">
     <c:redirect url="http://javatpoint.com"/>
  </c:if>
  <c:if test="${url>1}">
     <c:redirect url="http://facebook.com"/>
  </c:if>
</body>
</html>
```

JSTL(The JSTL function provides a number of standard functions, most of these functions are common string manipulation functions.)

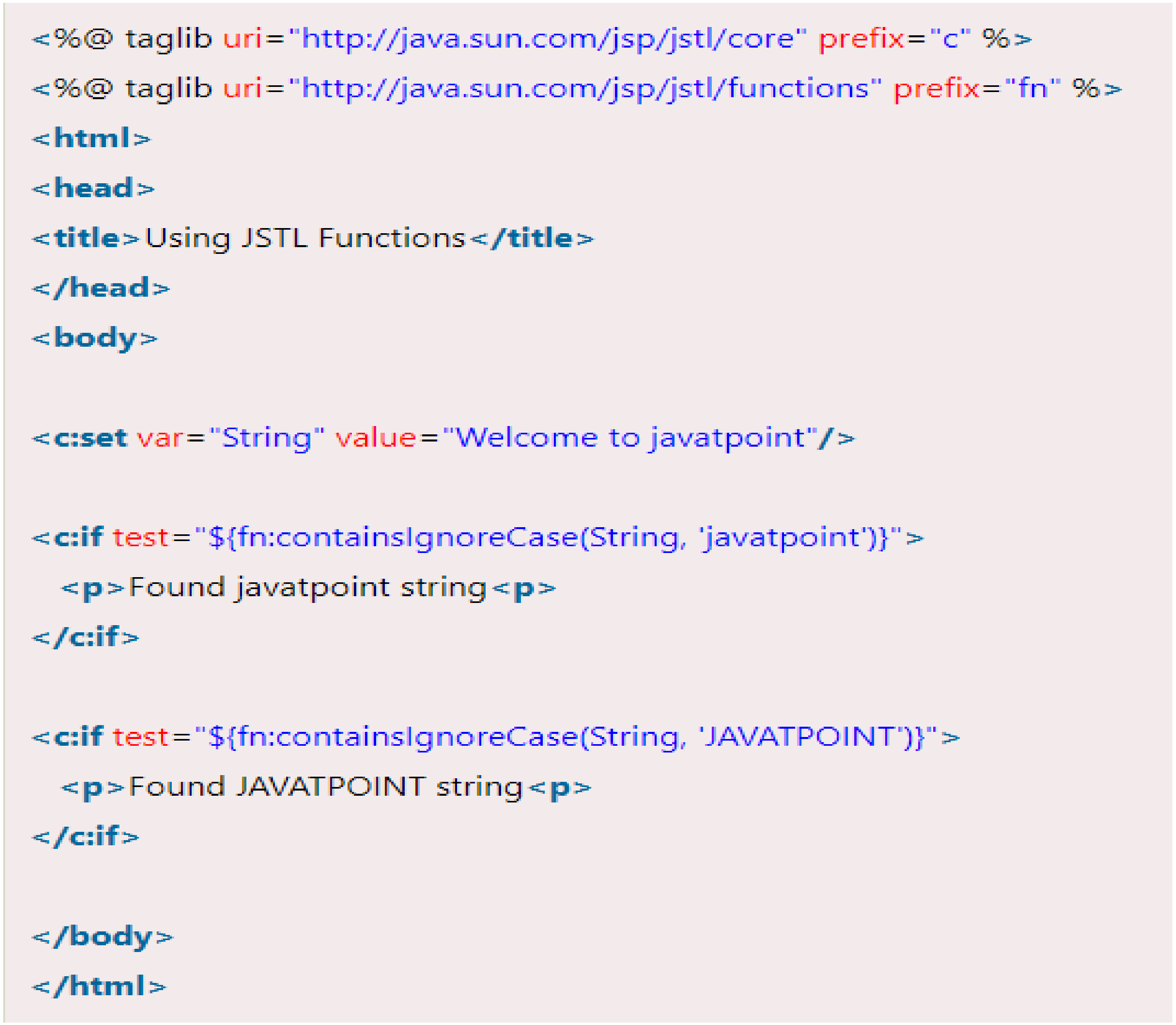
fn:contains()

Function



```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<head>
<title>Using JSTL Functions</title>
</head>
<body>

<c:set var="String" value="Welcome to javatpoint"/>

<c:if test="${fn:contains(String, 'javatpoint')}">
   <p>Found javatpoint string<p>
</c:if>

<c:if test="${fn:contains(String, 'JAVATPOINT')}">
   <p>Found JAVATPOINT string<p>
</c:if>

</body>
</html>
```

fn:containsIgnoreCase() Function

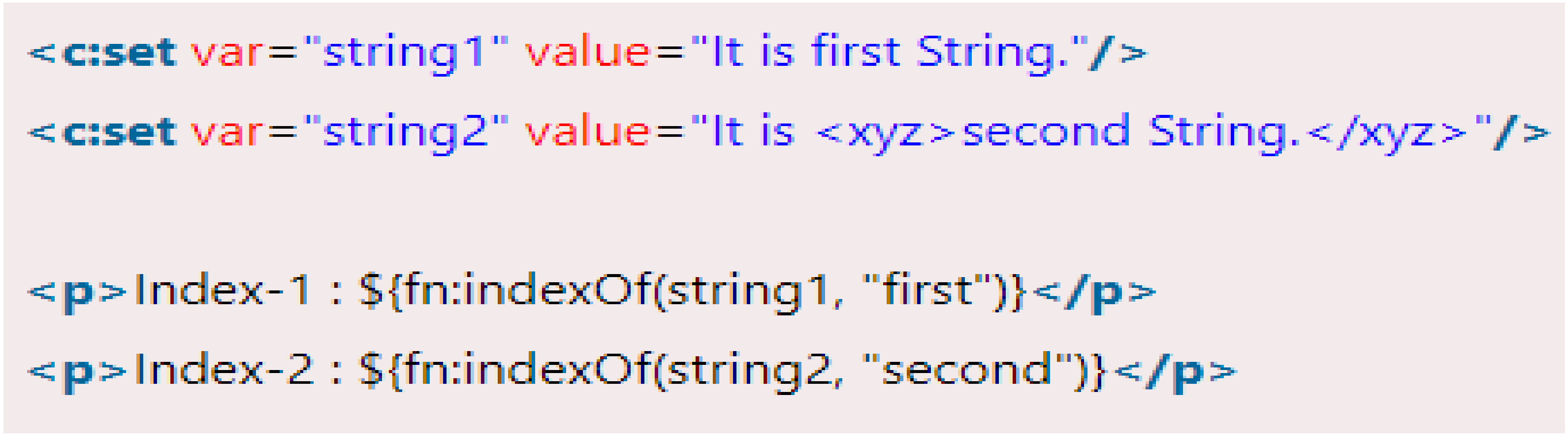


```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<head>
<title>Using JSTL Functions</title>
</head>
<body>

<c:set var="String" value="Welcome to javatpoint"/>

<c:if test="${fn:containsIgnoreCase(String, 'javatpoint')}">
   <p>Found javatpoint string<p>
</c:if>

<c:if test="${fn:containsIgnoreCase(String, 'JAVATPOINT')}">
   <p>Found JAVATPOINT string<p>
</c:if>

</body>
</html>
```

fn:endsWith() Function

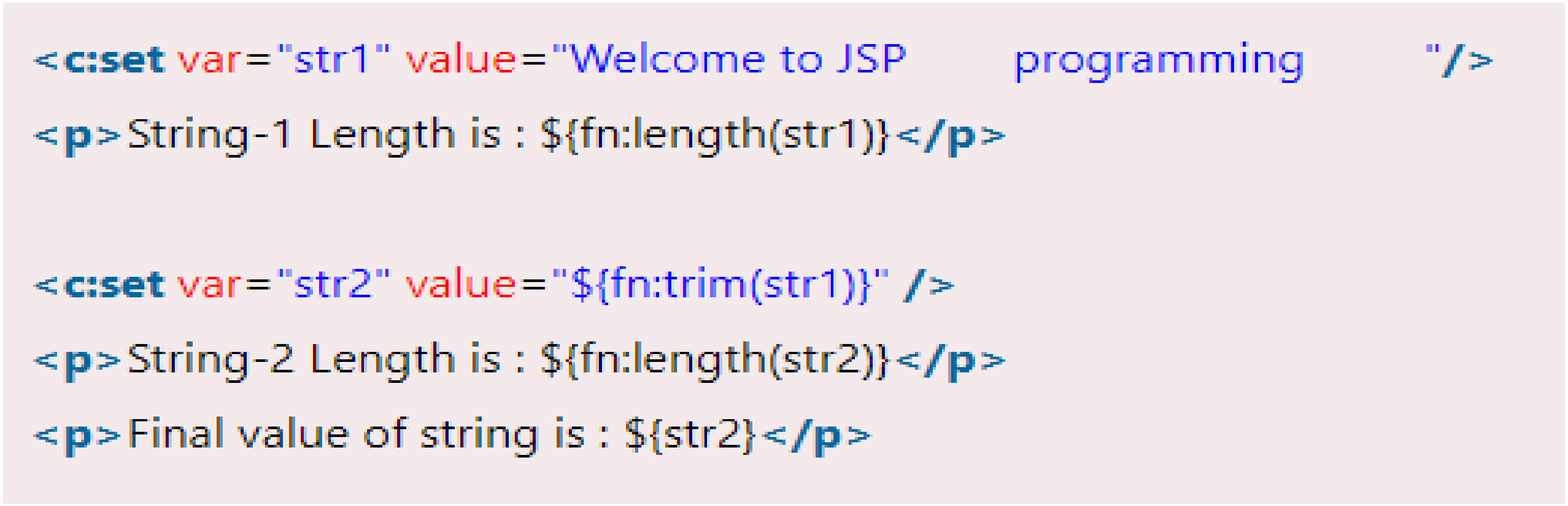
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<head>
<title>Using JSTL Functions</title>
</head>
<body>

<c:set var="String" value="Welcome to JSP programming"/>

<c:if test="${fn:endsWith(String, 'programming')}">
   <p>String ends with programming<p>
</c:if>

<c:if test="${fn:endsWith(String, 'JSP')}">
   <p>String ends with JSP<p>
</c:if>

</body>
</html>
```

fn:indexOf() Function



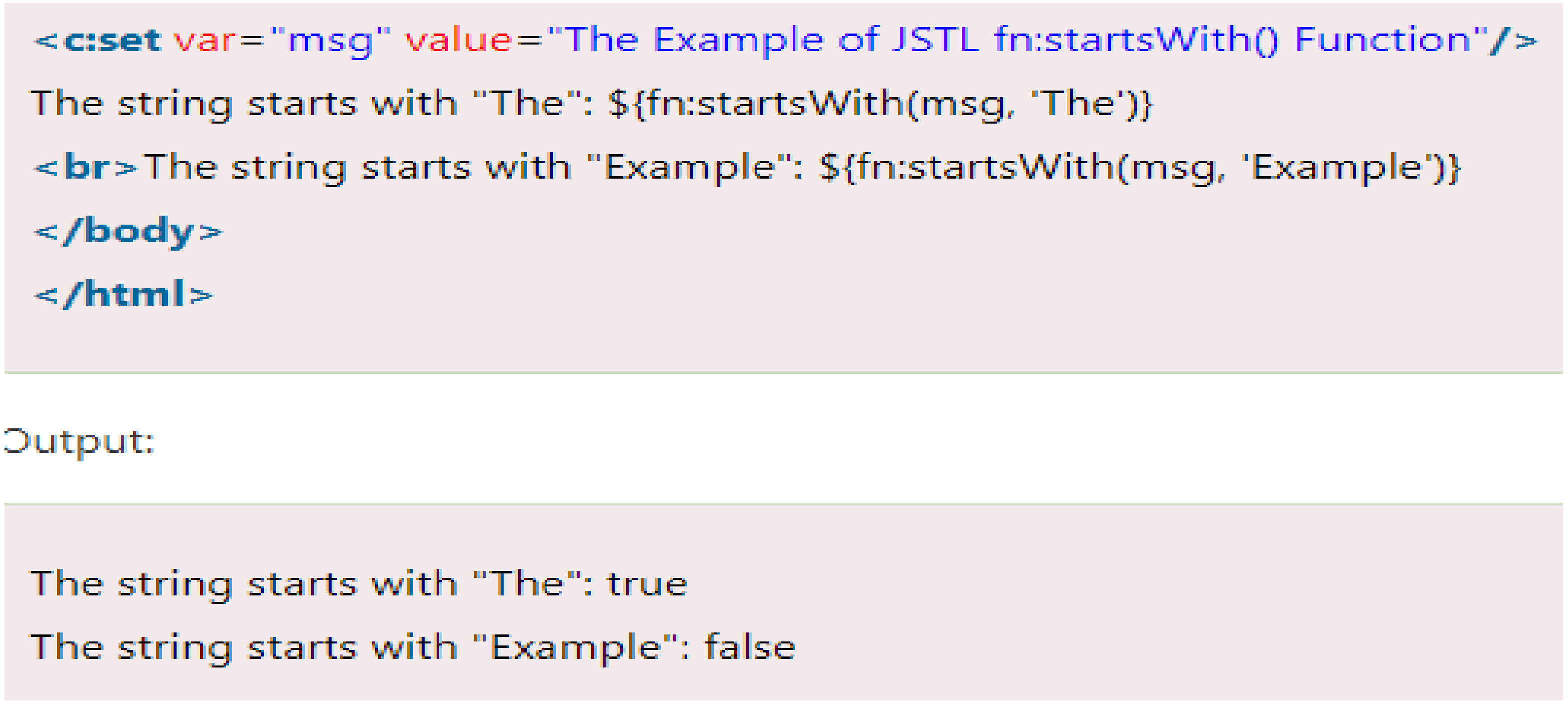
```
<c:set var="string1" value="It is first String."/>
<c:set var="string2" value="It is <xyz>second String.</xyz>"/>

<p>Index-1 : ${fn:indexOf(string1, "first")}</p>
<p>Index-2 : ${fn:indexOf(string2, "second")}</p>
```

fn:trim() Function



```
<c:set var="str1" value="Welcome to JSP       programming        "/>
<p>String-1 Length is : ${fn:length(str1)}</p>

<c:set var="str2" value="${fn:trim(str1)}" />
<p>String-2 Length is : ${fn:length(str2)}</p>
<p>Final value of string is : ${str2}</p>
```

fn:startsWith() Function

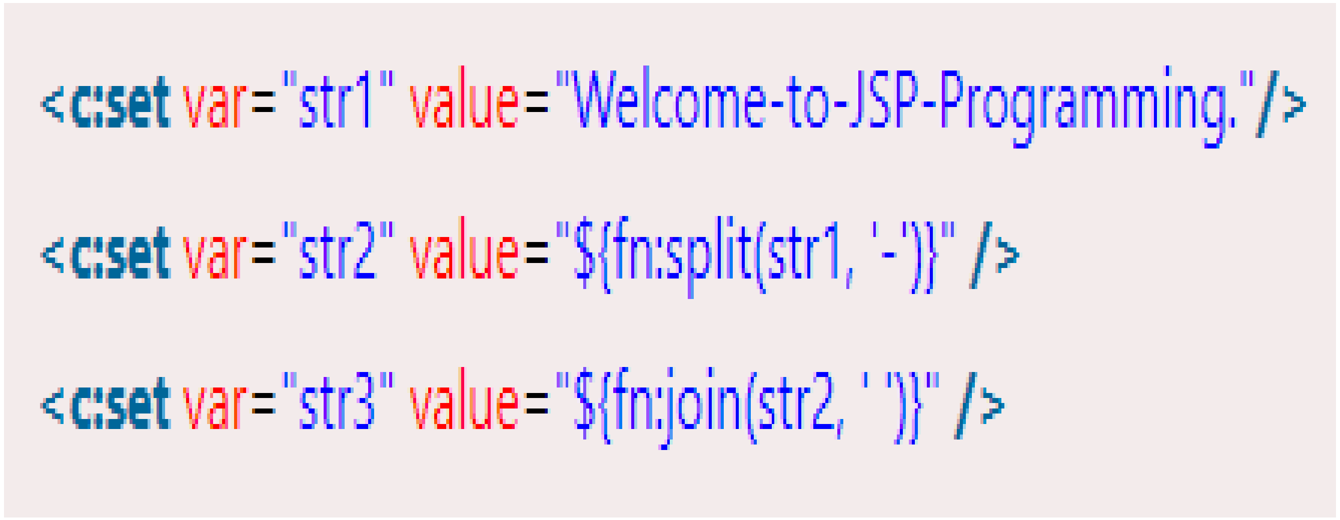


```
<c:set var="msg" value="The Example of JSTL fn:startsWith() Function"/>
The string starts with "The": ${fn:startsWith(msg, 'The')}
<br>The string starts with "Example": ${fn:startsWith(msg, 'Example')}
</body>
</html>
```

Output:

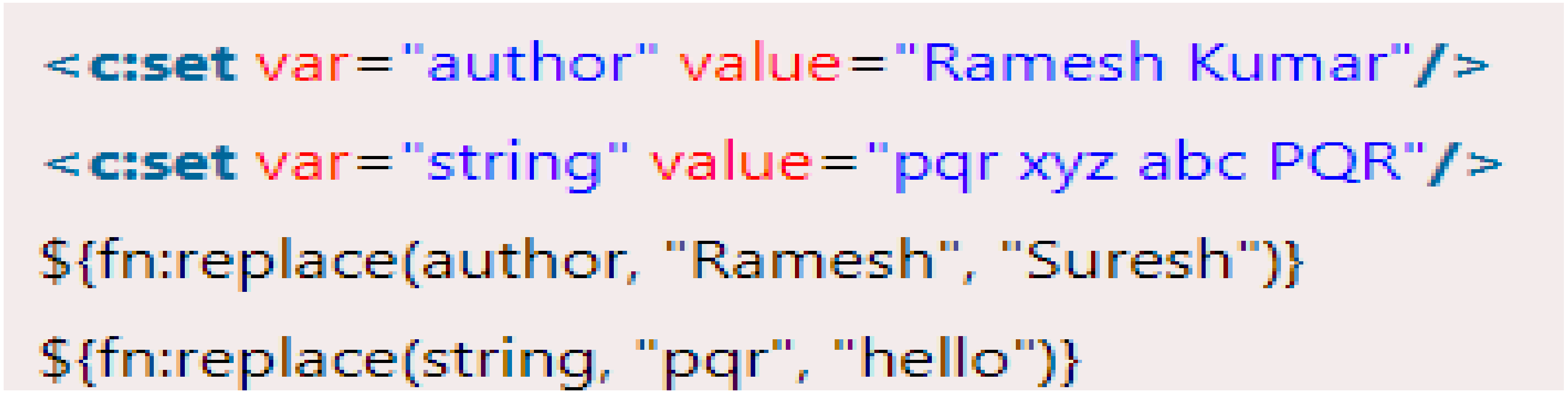
```
The string starts with "The": true
The string starts with "Example": false
```

- ## fn:split() Function



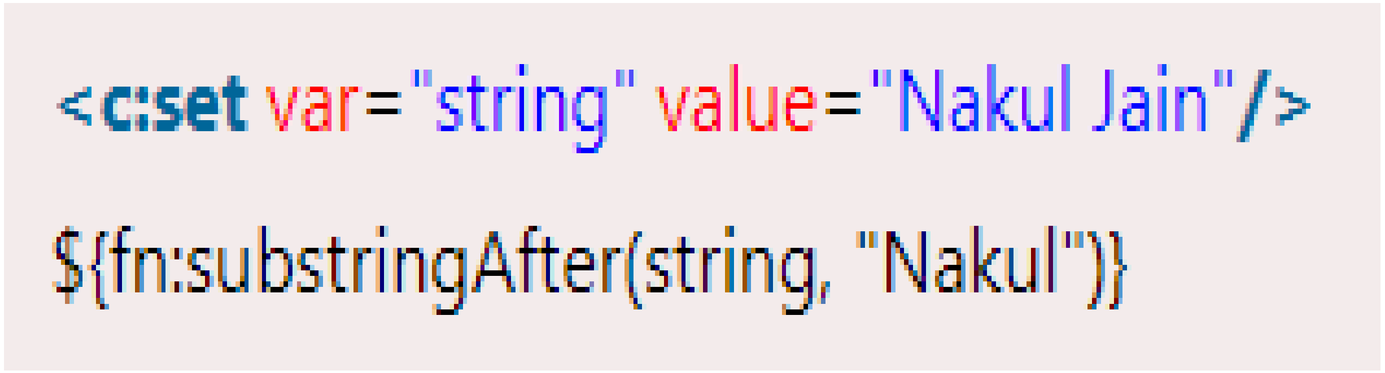
```
<c:set var="str1" value="Welcome-to-JSP-Programming."/>
<c:set var="str2" value="${fn:split(str1, '-')}" />
<c:set var="str3" value="${fn:join(str2, ' ')}" />
```

fn:replace() Function



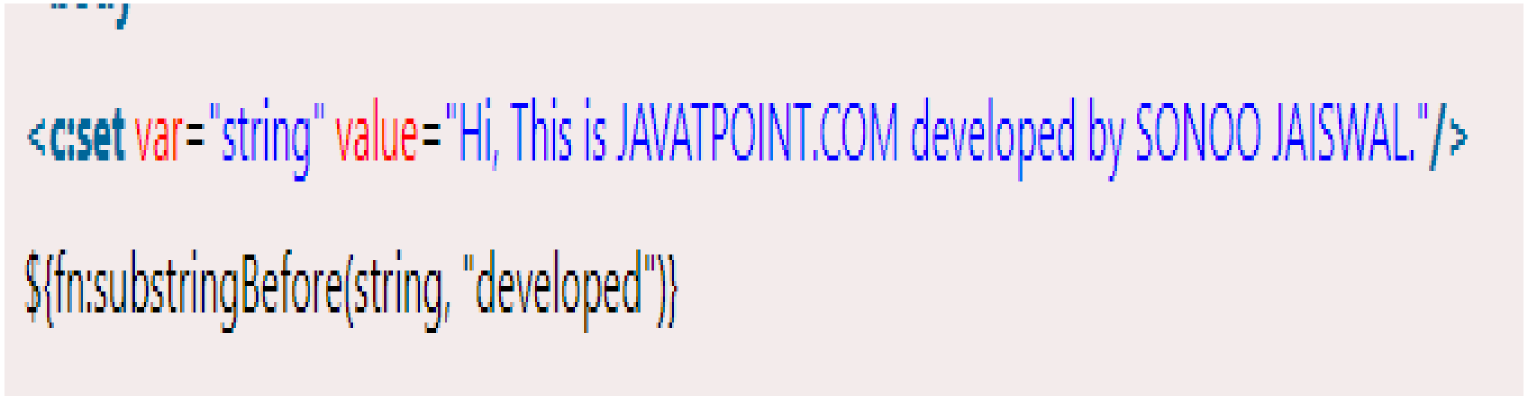
```
<c:set var="author" value="Ramesh Kumar"/>
<c:set var="string" value="pqr xyz abc PQR"/>
${fn:replace(author, "Ramesh", "Suresh")}
${fn:replace(string, "pqr", "hello")}
```

fn:substringAfter() Function



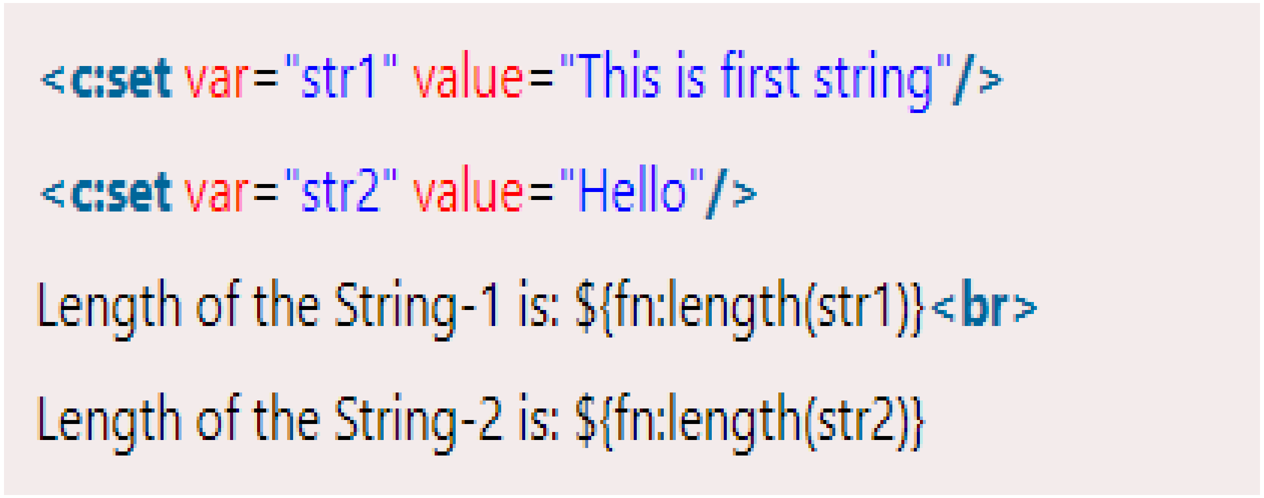
```
<c:set var="string" value="Nakul Jain"/>
${fn:substringAfter(string, "Nakul")}
```

fn:substringBefore() Function



```
<c:set var="string" value="Hi, This is JAVATPOINT.COM developed by SONOO JAISWAL."/>
${fn:substringBefore(string, "developed")}
```

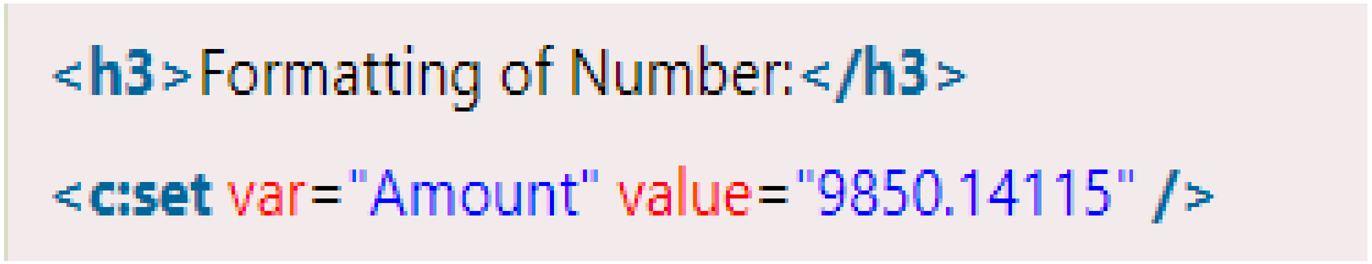
fn:length() Function



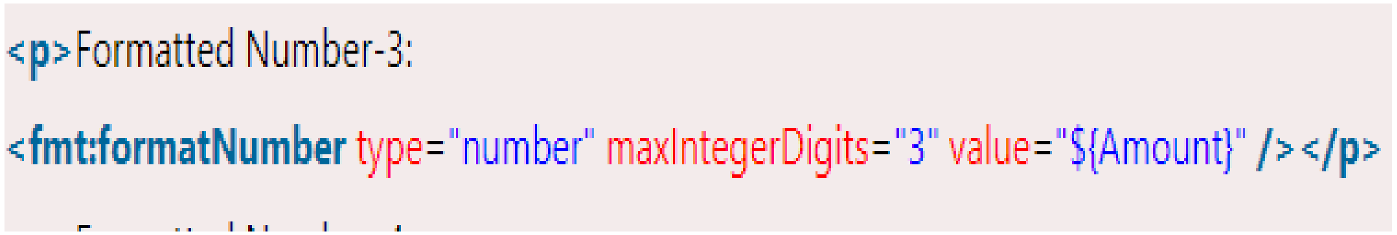
```
<c:set var="str1" value="This is first string"/>
<c:set var="str2" value="Hello"/>
Length of the String-1 is: ${fn:length(str1)}<br>
Length of the String-2 is: ${fn:length(str2)}
```

JSTL Formatting tags(The formatting tags provide support for message formatting, number and date formatting etc.)

<fmt:formatNumber> Tag

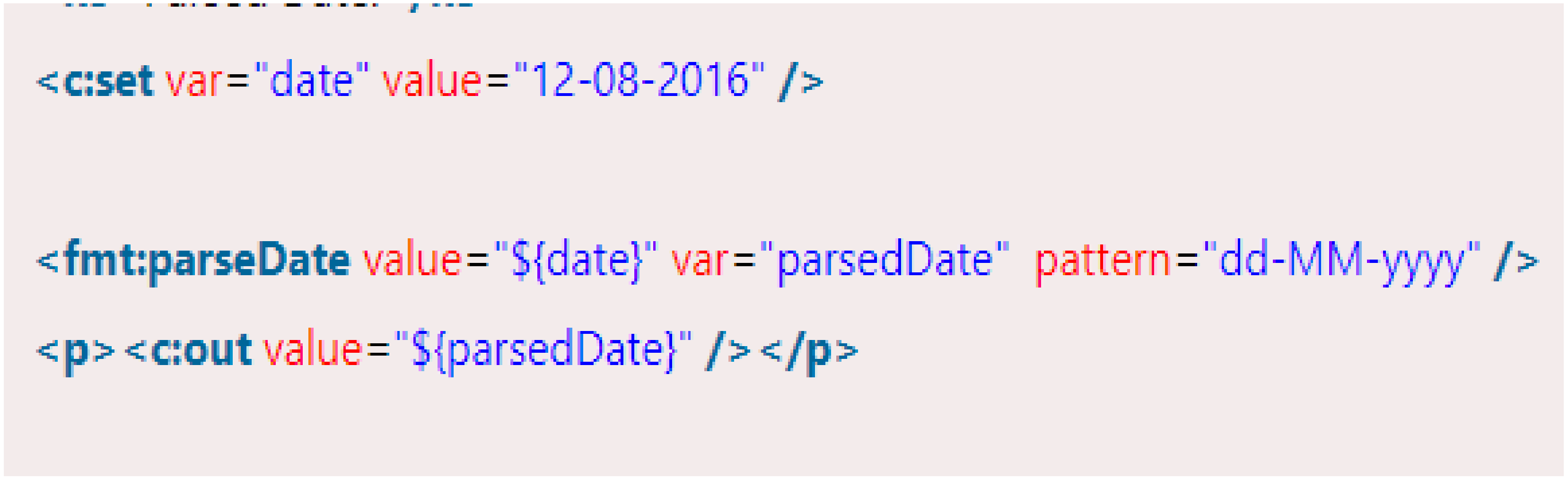


```
<h3>Formatting of Number:</h3>

<c:set var="Amount" value="9850.14115" />
```



```
<p>Formatted Number-3:

<fmt:formatNumber type="number" maxIntegerDigits="3" value="${Amount}" /></p>
```

<fmt:parseDate> Tag



```
<c:set var="date" value="12-08-2016" />


<fmt:parseDate value="${date}" var="parsedDate"  pattern="dd-MM-yyyy" />

<p><c:out value="${parsedDate}" /></p>
```