Assignment 1 :-

1. Write briefly about the different implicit objects in Jsp.

Ans:- There are 9 implicit objects. These objects are created by the web container that are available to all the Jsp pages.

1. Out :-
   - For writing any data to the buffer.
   - It is an object of JspWriter.
   In case of servlet we need to write:
   PrintWriter out = response.getWriter();
   - But in jsp, you don't need to write this code
   example:-
   <% out.println("welcome!"); %>

2. Request :-
   - It is an implicit object of HttpServletRequest.
   - It can be used to get request information such as parameter, header information etc.
   - It can be used to get, set and remove attributes from the jsp request scope.
   example:-
   index.html :-
   <form action = "welcome.jsp">
   <input type = "text" name = "uname">
   <input type = "submit" value = "go">
   </form>

welcome.jsp
<%
String name = request.getParameter("uname");

out.println ("welcome " + name);
%>

3) response :-
- In jsp, response is an implicit object of type HttpServletResponse. It is created by the web container for each jsp request.
- It can be used to add or manipulate response such as redirect response to another resource, send error etc.

example :-
welcome.jsp
<%
response.sendRedirect ("http://www.google.com");
%>

4. Config :-
- It is an implicit object of type. ServletConfig.
- It can be used to get initialization parameter for a particular jsp page.
- The config object is created by web container for each jsp page.
- It is used to get initialization parameter from the web.xml.

Example :-
web.xml :-
<init-param>
<param-name> dname </param-name>
<param-value> driver </param-value>
</init-param>
welcome.jsp :-
<% String driver = config.getInitParameter ("dname");
%>

5) JSP application

- It is an implicit object of type ServletContext.
- The instance of ServletContext is created only once by the web container when application or project is deployed on the server.
- It can be used to get initialization parameter from configuration file (web.xml). It can also be used to get, set or remove attribute from the application scope.

example:-

web.xml :-

```
<context-param>
    <param-name> dname </param-name>
    <param-value> sun.jdbc.odbc.JdbcOdbcDriver
                  </param-value>
</context-param>
```

welcome.jsp.

```
<%
    String driver = application.getInitParameter("d-
                    name");
%>
```

6. Session.

- In JSP, session is an implicit object of type HttpSession.
- It can be used to set, get or remove attributes or to get session information.

example:-

index.html:-

```
<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);
    session.setAttribute("user", name);
    <a href="second.jsp"> second.jsp </a>
%>
```

second.jsp

```
<%
    String name = (String) session.getAttribute("user");
    out.print("Hello "+name);
%>
```

7. pageContext.

- pageContext is an implicit object of type PageContext.
- It can be used to get, set or remove attribute from one of the following scopes:-

1) page
2) request
3) session
4) application

example:-

welcome.jsp:-

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

8. page.
- page is an implicit object of type Object class.
- This object is assigned to the reference of auto generated servlet class. It is written as:-

  Object page = this;
- For using this object it must be cast to Servlet type. For example:-

  <% (HttpServlet) page.log ("message"); %>
- Since, it is of type Object it is less used because you can use this object directly in jsp. For example:-

  <% this.log ("message"); %>

9. exception.
- exception is an implicit object of java.lang.Throwable class.
- It can be used to print the exception.
- But it can only be used in error pages.

Example:-

example.jsp

```
<%@ page isErrorPage = "true" %>
    Sorry following exception occured: <%= exception
%>
```

2. What is a jsp and the tags to work with the bean?

Ans:- JSP is a technology for developing web pages that support dynamic content which helps developers insert java code in html pages by making use of special JSP tags.

Using JSP, we can collect inputs from users

through web forms, present records from a database or another source, and create web pages dynamically.

Beans:-

- The jsp:UseBean action tag is used to locate or instantiate a bean class. If bean object of the bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.
- Syntax:-

<jsp:useBean id="instanceName" scope="page|request|session|application" class="packageName.className" type="packageName.className" beanName="packageName.className|<%= expression>" />

- Attributes:-

1. id:- is used to identify the bean in the specified scope
2. scope:- represents the scope of the bean. The default is page.
3. class:- instantiates the specified bean class but it must have no arg. constructor.
4. type:- provides the bean a data type.
5. beanName:- instantiates the bean using java.beans.Beans.instantiate method().

- jsp:setProperty and jsp:getProperty action tags:-

- The setProperty and getProperty action tags are used for developing web pages applications with java bean.

- In web development, bean class is mostly used because it is a reusable software component that represents data.
- The jsp:setProperty is sets a property value or values in a bean using the setter method.
- Syntax:-

```
<jsp:setProperty name = "instanceOfBean" property = "*"
property = "propertyName" param = "parameterName"
property = "propertyName" value = "<%=String
<%= expression %>" />
```

- The jsp:getProperty action tag returns the value of property.
- Syntax:-

```
<jsp:getProperty name = "instanceOfBean" property =
"propertyName" />
```

Example:-

TestBean.java

```
package action;
public class TestBean{
  private String message = " ";
  public String getMessage(){
    return (message);
  }
  public void setMessage(String message){
    this.message = message;
  }
}
```

main.jsp

```
<html>
  <head>
    <title> JavaBeans </title>
  </head>
  <body>
    <center>
      <h2> Using Javabeans </h2>
      <jsp:UseBean id = "test" class = "action.testBean" />
      <jsp:setProperty name = "test" property = "message"
      value = "Welcome" />
      <p> Got Message... </p>
      <jsp:getProperty name = "test" property = "message" />
    </center>
  </body>
</html>
```

5. What are the different scopes of the beans created? Explain with example?

Scopes:-

1. page:-
   - 'page' scope means, the jsp object can be accessed from with in the page only where it was created.
   - The default scope for jsp objects created using <jsp:Usebean> tage is page.

2. request:-
   - a jsp object created using the 'request' scope can be accessed from any page that serves the request.
   - More than one page can serve that request.
   - It will be ~~for~~ bound to the request object

- Session.
- Session scope means the jsp object is accessible from pages that belong to the same session from where it was created.
- This jsp object that is created using the session scope is bound to the session object

- application
- a jsp object created using application scope can be accessed from any page across the the application.
- The jsp object is bound to application object

example:-

welcome form:-

```
<html>
  <head>
    <title> Welcome Form </title>
  </head>
  <body>
    <form action = "useBean.jsp" method = "post">
      ID: <input type = "text" name = "id">
    Username: <input type = "text" name = "username">
    Email: <input type = "email" name = "email">
    Gender: <input type = "text" name = "gender">
      <button> Session These </button>
    </form>
  </body>
</html>
```

Usebean.java

```
package user;
public class Usebean{
    private int id;
    private String username, email, gender;
    public int getId(){
        return id;
    }
    public void setId(int id){
        this.id = id;
    }
    public string getUsername(){
        return username;
    }
    public void setUsername(String username){
        this.username = username;
    }
    public void setemail(String email){
        this.email = email;
    }
    public String getEmail(){
        return email;
    }
    public void getGender(){
        return gender;
    }
    public void setGender(String gender){
        this.gender = gender;
    }
}
```

UseBean.jsp :-

```
<body>
    <jsp:usebean id="vObject" class="User.Usebean"
    scope="session"/>
    <a href="sessionplace.jsp"> Click here </a>
    <jsp:setProperty property="*" name="vObject">
</body>
```

SessionPlace.jsp

```
<html>
    <head>
        <title> Infomation </title>
    </head>
    <body>
        <h1> Details are </h1>
        <jsp:usebean id="vObject" class="user.UseBean"
        session scope="session">
        <h3> UserID : <jsp:get Property property="id"
        name="vObject" />
        <h3> UserName: <jsp:getProperty property="
        username" name="vObject" />
    </body>
</html>
```

4. Difference between JSP include directive and jsp include action tag.

	Include Directive	Include Action
1.	It includes the file at translation time (the phase of jsp life cycle	It include file at runtime

where jsp gets converted into equivalent servlet

2. If the included file is changed but not the jsp which is including it then the changes will reflect not using when we are using include directive as the jsp is not changed. So it will not be translated.

The changes will be reflected only when we use jsp action tag.

3. Syntax:-
<%@ include file = "filename" %>

Syntax:-
<jsp:include page = "file name" />

4. In case of directives we cannot pass any parameter.

In action we can pass parameter
<jsp:include page = "file" />
<jsp:param name = "parameter name" value = "value" />
</jsp:include>

5. How are jsp's and servlets related

Ans:-

- JSP and servlets are both based on web technologies.
- Both can be used to process dynamic content on server and return it to client using a typical web browser.
- JSP is internally compiled as to a servlet. As jsp tags can be understood directly. So the jsp tags are first parsed and a servlet file is created during runtime and request to that jsp is then handled by this generated servlet.

6. Give the types of jsp scripting elements.

JSP scripting elements:-

- The scripting element lets the ability to insert java code inside the jsp.
- There are 3 types of scripting elements

1. Scriplet tag
2. Declarative tag
3. expression tag.

1. Scriplet tag:-

- It is used to execute java code in jsp.
- Syntax:-

<% java source code %>

example:-

```
<html>
  <body>
    <% out.println("Welcome to Jsp"); %>
  </body>
</html>
```

2. Expression Tag:-

- The code placed within JSP expression tag is written to the output stream of the response. So we need not write out.print() to write data. It is mainly used to print the values of variable or method

Syntax:-

<%= statement %>

Example:-

```
<html>
    <body>
        Current time = <%= java.util.Calendar.get
            Instance().getTime() %>
    </body>
</html>
```

3. Declaration Tag:-

- It is used to declare fields and methods
- The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.
- So it does not get memory at each request.

Syntax:-

<%! field or method declaration %>

example:-

```
<html>
<body>
<%!
    int cube(int n)
    {
        return n*n*n;
    }
%>
<%= "Cube of 3 is " + cube(3) %>
</body>
</html>
```

Assignment 2:-

1. Compare sendRedirect and forward method

	Forward	SendRedirect
1.	When we use forward method request is transfered to another resource within the same server for further processing	In case of sendRedirect() request is transfered to another resource to different or domain or different server for further processing
2.	Web container handle all process internally and client or browser is not involved.	When you use sendRedirect container transfers the request to client or browser so url given inside the method is visible as a new request to client
3.	When forward is called on request dispatcher object we pass request and response object so our old request object is present on new resource which is going to process our request.	In case of sendRedirect call old request and response object is lost because it is treated as new request by browser.

4.	Visually we are not able to see forward address its transparent.	In address bar we are able to see the new redirected address its not transparent.
5.	It is faster.	It is slow because. one extra round trip is required because completely new request is created and old request object is lost.
6.	When we redirect using forward and we want to use same data in new resource we can use request.setAttribute as we have request object available.	If we want to use or store the data in session or pass along the url.

2. Discuss advantages of using JSP over servlets

	JSP	Servlets
1.	Allows tag based programming. So extensive java knowledge is not required.	Does not allow tag based programming. So extensive java knowledge is required
2.	Suitable for both java & non java programmers	Not suitable for non java programmers

3. Use nine implicit objects which we can use directly in our program.	Implicit objects are present but we can't use them directly. We need write additional code to use them.
4. Modification done in JSP program will be recognized by underlying server automatically without reloading of web server / application.	Here we need to compile and reload manually.
5. Takes care of exception handling.	Does not take care of exception handling. Programmers have to explicitly handle this.
6. Allow us to use separate presentation logic (html code) from java code (business logic).	Does not allow.

3. Discuss the attributes of page directives

Page Directives:

- It provides attributes that gets applied to an entire page. It define page dependent attributes, such as scripting language error page and buffering req.
- It is used to provide instructions to container that pertains to current JSP page.

Syntax:-

<%@ page .... %>

Attributes:-

1. Language:- It defines the programming language being used in the page.
ex:-
<%@ page language = "java" %>

2. Extends:- This attribute is used to extend the classes like java does.
ex:-
<% page extends = "demotest.Democlass" %>

3. Import:- This attribute is mostly used attribute. It is used to tell the container to import other java classes, interfaces, enums etc while generating servlet code.
ex:-
<% page import = "java.util.Date" %>

4. content Type:-
. It defines the character encoding scheme i.e it is used to set the content type and the character set of the response. Default is "text/html"
ex:-
<% @ page contentType = "text/html" %>

5. info :- It defines a string which can be accessed by getServletInfo() method. It is used to get servlet description.
ex:-
<%@ page info = "Servlet Info" %>

6. Session:- JSP creates a session by default. Sometimes we don't want a session to be created. When it is set to false, then we indicate the compiler to not create a session by default
ex:-
<%@ page session = "true/false" %>

7. isThreadSafe:- It defines threading model for the generated servlet. It indicates the level of thread safety implemented in the page. It default value is true so simultaneous. We can use this attribute to implement single thread model
ex:-
<%@ page isThreadSafe = "true/false" %>

8. Autoflush:- This attribute specifies that the buffered outputs should be automatically flushed or not default value is true
ex:-
<%@ page autoFlush = "true/false" %>

9. Buffer:- The value represents the size of buffer. If there is no buffer then we can write as none. default is 8Kb.
ex:-
<%@ page buffer = "16KB" %>

10. isErrorPage:-
- It indicates that jsp page that has an error page will be checked in another jsp.
- Exceptions are available to this page only.

- default is false
  ex:-
  <%@ page isErrorPage = "true" %>

11. PageEncoding :- It defines character encoding for JSP page. The default is "ISO-8859-1"
  ex:-
  <%@page pageEncoding = "ISO-8859-1" %>

12. ErrorPage:- It is used to set error page for the jsp if throws an exception.
  ex:-
  <%@ page errorPage = "File1.jsp" %>

13. isELIgnored:- It is flag attribute when we have to decide whether to ignore EL tags or not
  ex:-
  <%@ page isELIgnored = "true/false" %>

4. Compare jsp:forward with jsp:include.

	Property	include	forward
1.	Occurs	At translation time	At request time
2.	Performance	comparatively slower	faster
3.	No. of servlets created	Only one	multiple

	Property	include	forward
4.	Response data sent	Includes both jsp including and included	Goes only that of included jsp.
5.	Used when	When content of included file does not change often like advertisement banners.	When content of included file changes often.
6.	Tranfer of control	Execution control shifts temporarily to included file when it is exe. It transfers back.	Shifts permanently to the included file.
7.	Response to clients	Goes from same jsp which client requested	Goes from deff client jsp.
8.	Merge of response	Both of including & included file.	No merging.
9.	Extra activity	When control returned including of other file can be done	No extra activity.

5. Explain jsp:fallback ?

Syntax:-

<jsp:fallback> text message for user </jsp:fallback>

- A text message to display for the user if the plug-in cannot be started.
- If plug-in starts but the applet or Bean does not, plug-in usually displays pop-up window explaining the error to user.
- Its used with <jsp:plugin> element

Example:-

```
<jsp:plugin type="applet" code="Clock2.class".
  codebase="applet" jreversion="1.2" width="160"
height="150">
<jsp:fallback>
      plugin tag object or embed not supported
by browser
</jsp:fallback>
</jsp:plugin>
```

6. What is bean? Explain in detail?

- A java Bean is a class that should follow the following conventions.

1. It should have a no-arg constructor.
2. It should be serializable
3. It should provide method to get and set the value of properties, known as getter setter methods.

How to access java Bean class

```
package mypack;
public class Test {
    public static void main (String args[]){
        Employee e = new Employee (); // object is
                                      // created
        e.setName ("Antra"); // setting value to the object
        System.out.print (e.getName());
    }
}
```

Advantages:-

- Java Bean property and methods can be exposed to another application.
- It provides an easiness to reuse software components.

Disadvantages:-

- JavaBeans are mutable..
- Creating getter & setter method for each property separately may lead to the boilerplate code.

7. What are jsp directives and how are they different from scripting elements?

- Jsp directives are message to JSP container. They provide global information about an entire jsp page.
- Jsp directives are used to give special instructions to a container for translation of Jsp to servlet code.
- In Jsp life cycle phase, Jsp has to be converted to a servlet which is the translation phase.

- They give instructions to the container on how to handle certain aspects of jsp page processing.
- Directives can have many attributes by comma separated as key-value pairs.
- In jsp directive is described in <%@ %> tags

Syntax:-

<%@ directive attribute = "" %>

Types:-

1. Page
2. Include
3. Taglib.

Compare with Scripting elements.

Scripting elements provides the ability to insert java code inside jsp. whereas the jsp directives tell the container on how to actually handle different aspects of jsp page.

Assignment 3:-

Q1:- What are the steps for implementing MVC?

Ans:- Steps:-

1. Define beans to represent data.
2. Use a servlet to handle request.
3. Populate the beans. The servlet invokes the business logic or data access code to obtain results. The results are placed in the beans that were designed in step 1.
4. Store the bean in the request, session or servlet context. The servlet calls .setAttribute on the request, session or servlet context objects to store the reference to the beans that represent the result of a request.
5. Forward the request to a jsp page. The servlet determines which jsp page is appropriate to the situation and uses the forward method of RequestDispatcher to transfer control to that page.
6. Extract the data from the beans. The jsp page access bean with jsp:useBean and a scope matching the location of step 4. The page then uses jsp:getProperty to output the bean properties. The jsp page does not create or modify the bean, it merely extracts and displays the data that the servlet created.

Q2 Why MVC framework is required to build complex application?

Ans:-

Saves Time & effective use of Resources.

Due to the separation of components. MVC allows the reuse of business logic across platforms. In addition, multiple user interface can be developed in line without concerning. the codebase. Two different programmers can work simultaneously on two different business logic. This makes the work faster, saves time, and helps in to manage the resources effectively.

Facilitates multiple views.

Due to the advantage of working on separate data and different business logic, duplication of code is certainly less. The separation of view model enables the user interface to display multiple views of the same data at the same time.

Modification does not effect entire model.

Change is part of life when it come to web application development changes become part of the ongoing development process. When it comes to a UI that changes are frequent may it be a change in colour. In addition UI updating can be made without slowing down business logic

SEO friendly platform

Web development and SEO go hand in hand MVC platform supports the development of SEO friendly applications. It provides ease to develop SEO friendly URL's in order to generate more visits on a specific page.

Q3 Discuss the operators available in EL.

Ans- JSP Expression Language (EL) supports most of the arthimetic and logical operators supported by java.

S.NO	Operator & Description.
1.	. Access a bean or Map entry
2.	[] access an array or list element
3.	() Group a subexpression to change the evaluation order
4.	+ Addition.
5.	- Subtraction
6.	* Multiplication

7. / or div
Division

8 % or mod
Modulo

9 == or eq
Test for equality

10 != or ne
Test for inequality

11 < or lt
Test for less than

12 > or gt
Test for greater than

13 <= or le
Test for less than or equal

14. >= or ge
Test for greater than or equal

15 && or and
Test for logical AND

16. || or or
Test for logical OR

17. ! or not
Unary Boolean compliment

18 empty
Test for empty variable values

4. How can you access a scoped variable using expression language? Example

Ans:-

1. PageScope :- Scoped variables from page.
2. requestScope :- Scoped variables from request scope
3. sessionScope :- Scoped variables from session scope
4. applicationScope :- Scoped variables from application scope.

The pageScope, requestScope, sessionScope and application Scope variables provide access to variables stored at each scope level.

Application
SessionScope Example :-

In this example, attributes have been set using application implicit object and on the display page we have got those attributes using applicationScope of EL.

index.jsp.

```
<html>
    <head>
        <title> EL example </title>
    </head>
    <body>
        <%
            application.setAttribute("name", "Antra")
```

```
        application.setAttribute("rollno", "134");
    %>
    <a href = "display.jsp"> Click </a>
  </body>
</html>
```

display.jsp

```
<html>
  <head>
    <title> Display Page </title>
  </head>
  <body>
    ${applicationScope.name} <br>
    ${applicationScope.rollno}
  </body>
</html>
```

5. How can you evaluate expression conditionally with EL?

1. "Jsp if-else."

"If else" is basic for all control flow statements, and it tells the program to execute the certain section of code only if the particular test evaluates to true.

- If the first condition is true then "if block" is executed and
- If it is false then "else block" is executed.

Syntax:-

```
If (test condition)
{
    // Block of statements
}
else
{
    // Block of statements
}
```

Example:-

```
<html>
  <head>
    <title> Conditional if-else </title>
  </head>
  <body>
    <%! int month = 5; %>
    <% if (month == 2) { %>
      <a> Its february </a>
    <% } else { %>
      <a> Any month </a>
    <% } %>
  </body>
</html>
```

2. Jsp-switch

The body of the switch ~~block~~ statement is called "switch block"

- It is used to check the no. of execution paths
- It can be used with all data types.
- It contains more than one cases and 1 default.
- It evaluates the exp. then executes all the statements following the matching case.

- Syntax:-

```
switch (operator)
{
    case1 :
        Block of statement.
    break;
    case 2:-
        Block of statement
    :
    default
        Block of statements
    break;
}
```

Example:-

```
<html>
    <head>
        <title> Switch </title>
    </head>
    <body>
    <%! int week = 2; %>
    <% switch (week) {
      case 0 :
          out.println ("Sunday");
          break;
      case 1;
          out.println ("Monday");
          break;
      case 2:
          out.println ("Tuesday");
          break;
      default :
          out.println ("Saturday");
    }
```

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

Q6:- Write the jsp comment syntax?-

Ans:- JSP comments marks to text or statements that the jsp container should ignore.

Syntax:-

```
<%-- This is a jsp comment --%>
```

Example:-

```
<html>
    <head>
        <title> A Comment </title>
    </head>
    <body>
        <h2> A Test of Comments </h2>
        <%-- This is comment --%>
    </body>
</html>
```

Q7:- Explain the attributes of getProperty and setProperty tag of jsp?

Setproperty:-

- Sets a property value or values in a bean.

Syntax:-

```
<jsp:setProperty
```

name = "beanInstanceName" . property = "*" |
property = "propertyName" [param = "parameter value"]
| property = "propertyName" value = {string |
<%= expression %> " />

Attributes:-

1. name:- The name of the Instance of a bean that has already been created or located with <jsp:useBean> tag. The value of name must be same as the value of id in <jsp:useBean>. The <jsp:useBean> tag must be used before <jsp:setProperty>

2. property = "*" :- Stores all the values in the request object parameters in the matching bean properties. The property names in the bean must match the request parameters.

3. property = "propertyName" [param = "value"].
Sets one Bean property to the value of request parameter. The request parameter can have a different name than Bean property.

4. property = "propertyName" value = "{string | <%= expression %> }"
Sets one Bean property to a specific value. The value can be string or expression.

- <jsp:getProperty>
  Gets the value of a Bean property so that we can display in result page.

Syntax:-

<jsp:getProperty name = "beanInstanceName" property = "propertyName" />

Attributes:-

1. name = The name of instance.
2. property = The name of property whose value we want to get.

Assignment-4

1. How to access nested bean and collection in JSP using JSTL?

Dot Operator:-

Also known as property access operator. This operator allows us to access property of any bean using dot notation.

For example:- To access the value of operator.property with name.property-name of bean, we can use below syntax.

${ bean.property-name }

There can be multiple level of nesting. EL allows to add scope to get the bean of a particular scope like ${requestScop.bean property }. If we do not add scope, EL searches for the bean in the order of page, request, session, application

Collection Access Operator:-

EL supports [] operator which can be used to get the data from array and list along with beans.

For example:-

${requestScope["a.b"]}
${mylist[1]} or ${mylist["1"]}

2. How to access body content of tag in JSTL?
For example:-
If our custom tag is xyz then we have to access the content between

<prefix:xyz>
Body of tag
</prefix:xyz>

Details.java

```
package detail.com;
import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;
public class Details extends SimpleTagSupport {
    StringWriter sw = new StringWriter();
    public void doTag throws JspException, IOExceptio
    {
        getJspBody().invoke(sw);
        JspWriter out = getJspContext().getOut();
        out.println(sw.toString() + "Appended Custom
            tag message");
    }
}
```

message.tld:

```
<taglib>
  <tlib-version> 1.0 </tlib-version>
  <jsp-version> 2.0 </jsp-version>
  <short-name> My Custom Tag : MyMsg </short-name>
  <tag>
    <name> MyTagMsg </name>
    <tag-class> detail.Detail </tag-class>
    <body-content> scriptless </body-content>
  </tag>
</taglib>
```

index.jsp

```
<%@ taglib prefix = "myprefix" uri = "WEB-INF/
    message.tld" %>
<html>
  <head>
    <title> Body Content access </title>
  </head>
  <body>
    <myprefix : MyMsg>
        Test String
    </myprefix : MyMsg>
  </body>
</html>
```

Assignment-5

Q8) Explain c:forEach, c:import and c:url with all the attributes.

Ans:-

<c:forEach>

- This is used for executing the same set of statements for a finite number of times.
- It is similar to for loop in java.
- This is basically used for when we need to perform (execute) set of statements again and again for a specified number of times

Syntax:-

```
<c:forEach var="counter" begin="initial-value"
    end="final-limit">
        // Block of statement
</c:forEach>
```

Attributes:-

Begin: The initial counter value
end: The final limit till which the loop executes
var: counter variable name.

Example:-

```
<html>
    <head>
        <title> c:forEach </title>
    </head>
    <body>
        <c:forEach var="counter" begin="1" end="10">
            <c:out value="${counter}" />
        </c:forEach>
    </body>
</html>
```

<c:import>

- It is used for importing the content from another file/page to the current JSP page.

Syntax:-

```
<c:import var="variable_name" url="relative_url"/>
```

Attribute:-

- url:- It is the url for address of file/page which needs to be import
- var:- It is the variable which stores the data imported from another url.

Example:-

```
<html>
    <head>
        <title> c:import </title>
    </head>
    <body>
        <c:import var="mydata" url="/display.jsp">
            <c:out value="${mydata}" />
        </c:import>
    </body>
</html>
```

<c:url>

- It is used for url formatting. It is mainly used when we want to open a jsp page based on the user input or based on the value of a variable.

Syntax:-

```
<c:url value="/file1.jsp" />
```

Attributes:-

- var:- variable name to store the formatted url.
- context:- used for specifying the application.
- scope:- The scope in which the var attribute would be stored. It can be request, page, application or session.

Example:-

```
<html>
    <head>
        <title> c:url </title>
    </head>
    <body>
        <c:url value = "file1.jsp" />
    </body>
</html>
```

2. What is javascript? Explain its features.

Javascript:-

- It is a programming language for web.
- It can update and change both HTML and CSS
- It can calculate, manipulate and validate data.

Features:-

- It is object based scripting language.
- Giving the user more control over the browser.
- It handles date & time.
- It can detect users browser and OS
- It is light weighted.
- It is a scripting language and it is not java
- It is interpreter based scripting language.
- It is case sensitive.
- It provides pre-defined objects.

- Every statement in javascript must be terminated with semi colon

3. Describe advantages and disadvantages of javascript.

Ans:- Advantages :-

1. Speed. Client side javascript is very fast because it can run immediately within the client-side browser. Unless outside resources are required, Javascript is unhindered by network calls to backend server.
2. Simplicity. Javascript is relatively simple to learn and implement.
3. Popularity. Javascript is used everywhere on web.
4. Interoperatability. Javascript plays nicely with other language and can be used in a huge variety of applications.
5. Server load. Being client-side reduces the demand on the website server.
6. Gives the ability to create rich interfaces.

Disadvantages :-

1. Client side security. Because the code extends on the user's computer, in some cases it can be exploited for malicious purposes. This is one reason some people choose to disable Javascript
2. Browser support. Javascript is sometimes interpreted differently by different browsers. This makes it somewhat difficult to write cross browser code.

4. How to create user defined object in javascript? Explain with example.

Ans:- Creating objects in javascript:-

1. By object literal
2. By creating instance of object directly (using new keyword)
3. By using an object constructor. (using new keyword).

1. By using object literal.

Syntax:-

```
object = { property1 : value1 , property2 : value2 ...
         ... propertyN : valueN }
```

Example:-

```
<script>
    emp = {id : 102, name : "Shyam Kumar",
           salary : 40000 };
    document.write (emp.id + " " + emp.name +
                    " " + emp.salary);
</script>
```

2. By creating instance of object

Syntax:-

```
var objectname = new Object();
```

Example:-

```
<script>
    var emp = new Object();
    emp.id = 101;
    emp.name = "Ravi Malik";
```

```
emp.salary = 50000;
document.write(emp.id+" "+emp.name+" "+emp.
        salary);
</script>
```

3. By using constructor.

Syntax:-

```
Object name = new object (values....)
```

- Here we need to create function with arguements. Each arguements value can be assigned in the current object by using this keyword.
- This keyword refers to current object.

Example:-

```
<script>
function emp (id, name, salary){
    this.id = id;
    this.name = name;
    this.salary = salary;
}
e = new emp (101, "Antra Kaul", 30000);
document.write (e.id+" "+e.name+" "+e.salary);
</script>
```

5. Difference between

	Confirm Dialog box	Prompt Dialog box
i.	It displays a pop up message with "Ok" & "cancel" buttons.	prompt enables to take user's input with "Ok" and "cancel" buttons.

	Confirm Dialog Box.	Prompt Dialog Box
2.	It is used to users confirmation to proceed.	It returns value entered by user. It returns null if the user does not provide any input value.

	Include Directive	Include Action
1.	It includes file at translation time. (the phase of jsp life cycle where the jsp gets converted into the equivalent servlet.	It includes file at runtime.
2.	If the included file is changed but not the jsp which is including it then the changes will not be reflected.	The changes will only be reflected if we use include action.
3.	Syntax:- <%@ include file = "file-name" %>	Syntax:- <jsp:include page = "file-name" />
4.	It is not possible to pass parameters.	While using jsp-action include we can also pass parameter.

6. Define jstl. State types of JSTL and dicuss c:forTokens, c:redirect, c:catch, c:import and c:param.

Ans:- JSTL:-

- It is a collection of useful JSP tags which encapsulates the core functionality common to many JSP applications.
- It has support for common, structural tasks etc.

Types:-

- Core Tags
- Formatting Tags
- SQL Tags
- XML Tags
- JSTL functions

1. <C:forTokens>

- Tag iterates over tokens which is separated by supplied delimiter. It is used to break string into tokens and iterate through each of tokens to generate output.

Example:-

```
<c:forTokens items="Rahul-Nakul-Rayesh" delims="-" var="name">
    <c:out value="${name}"/><p>
</c:forTokens>
```

2. <c:redirect>

- Tag redirects the browser to new url. It supports the context-relative URLs, and the <c:param> It is used for redirecting the browser to an alternate URL by using automatic URL rewriting

Example:-

```
<c:set var="url" value="0" scope="request" />
<c:if test="${url<1}">
    <c:redirect url="http://google.com" />
</c:if>
```

3. <c:catch>

- It is used to catch any Throwable exception that occurs in the body and optionally exposes it. In general it is used for error handling.

Example:-

```
<c:catch var="catchTheException">
    <% int x = 2/0; %>
</c:catch>

<c:if test="${catchTheException != null}">
    <p> The type of exception is : ${catchTheException}
</c:if>
```

4. <c:import>

- It is used for including the content of any resources either within the server or outside the server.
- It provides all functionality of include action.

Example:-

```
<c:import var="data" url="http://www.google.com" />
```

5. <c:param>

- It adds the parameter in a containing 'import' tag's URL. It allow the proper URL request parameter to be specified within URL and it automatically perform any necessary URL

encoding.

Example:-

```
<c:url value = "/index1.jsp" var = "completeURL" />
  <c:param name = "trackingId" value = "706" />
  <c:param name = "user" value = "Antra" />
</c:url>
```

7. Explain JSP page directives import, session, buffer and errorPage attributes with example:-

1) Import:-

- This attribute is most used attribute in page directive.
- It is used to tell the container to import other java classes, interfaces, enums etc. while generating Servlet code. It is similar to import statement in java classes, interfaces.

Syntax:-

```
<%@ page import = "value" %>
```

Examples:-

```
<%@ page import = "java.util.Date" %>
```

2) session.

- jsp creates session by default.
- Sometimes we dont want a session to be created in jsp and hence, we can set this attribute to false in that case. The default value of the session attribute is true

Syntax:-

<%@ page session = "true/false" %>

Example:-
<%@ page session = "false" %>

3. Buffer.
- Using this attribute the output response object may be buffered.
- We can define the size of buffering to be done using this attribute and default size is 8KB.
- It directs the servlet to write the buffer before writing to the response object

Syntax:-
<%@ page buffer = "value" %>

Example:-
<%@ page buffer = "16KB" %>

4. errorPage.
- This attribute is used to set the error page for the jsp page if jsp throws an exception and then it redirects to the exception page.

Syntax:-
<%@ page errorPage = "value" %>

Example:-
<%@ page errorPage = "errorHandler.jsp" %>