JSP------> To attract frontend developers to build webapplication without having knowledge of java
They should be comfortable only with presentation skills.

page directive attributes
==========
1. import
2. session
3. contentType
4. buffer
5. autoFlush
6. isELIgnored
7. errorPage
8. isErrorpage
9. info
10 .language

3. contentType
<%@ page contentType='application/pdf'%>
We can use this attrribute to set the response type(MIME type of response)
Default value is : text/html

4. isELIgnored
Inside jsp page if we want to use modern elements syntax (EL) syntax , then we need to use
isELIgnored attribute.
1. <%@ page isELIgnored = 'true' %>
EL syntax won't be processed and just treated as plain text.
2. <%@ page isELIgnored = 'false' %>
El syntax will be processed and prints its value.

eg::
```
<%@ page language ="java" isELIgnored ='false'%>
<h1>Working with Page Directives...</h1>
<h1>
      Username is :: <%= request.getParameter("username") %>
</h1>
<h1>
      Accessing Username through EL Syntax :: ${param.username}
</h1>
```

output
=====
http://localhost:9999/FirstApp/index.jsp?username=sachin
Working with Page Directives...
Username is :: sachin
Accessing Username through EL Syntax :: sachin

Note:
if isELIgnored = 'true', then EL won't be evaulated it would be treated as a plain text
Default value of isELIgnored = false.
Output
======
http://localhost:9999/FirstApp/index.jsp?username=sachin
Working with Page Directives...
Username is :: sachin
Accessing Username through EL Syntax :: ${param.username}

5. info

```
	<%@ page info = "Application developed by iNeuron" %>
	 To get this value we use a method called "getServletInfo()".
	 The default value of info is "Jasper JSP 2.3 Engine".
```

eg:

```
<%@ page language ="java" info = "Application developed by iNeuron"%>
<h1>Working with Page Directives...</h1>
<h1>
	<%= getServletInfo() %>
</h1>
```

6. buffer,autoFlush

```
	<%@ page buffer='52kb' %>
	The default value of buffers is 8kb.
```

autoFlush ->It is a boolean attribute which is used to give an information to the container about flushing the dynamic response to the client automatically or not.

if autoFlush value is true, then container will flush the complete response to the client from the buffer when it reaches the maximum capacity.

if autoFlush value is false, then container will raise an exception when the buffer is filled with the response.

eg:

```
<%@ page language ="java" buffer='52kb' autoFlush='true'%>
<h1>Working with Page Directives...</h1>
<%
	for(int i = 0;i<=100000000;i++)
		out.println("iNeuron");
%>
```

7. errorPage,isErrorPage

errorPage -> if any exception occurs in the jsp page then we need to forward the exception object to other jsp page, to do so we need to use the attribute called "errorpage".

isErrorPage-> It helps the jsp container to allow exception object inside the jsp page or not.

By default the value is 'false', so implicit object exception is not allowed inside jsp page.

To make it accessible inside jsp page we need to use 'true' value.

index.jsp

```
<%@ page language ="java" errorPage = 'error.jsp'%>
<h1>Working with Page Directives...</h1>
<%
	java.util.Date d =null;
	out.println(d.toString());
%>
```

error.jsp

```
<%@ page language ='java' isErrorPage='true'%>
<html>
	<body bgcolor='cyan'>
```

```
            <center>
                  <b>
                        <font size='5' color='red'>
                              <%=exception%>
                        </font>
                  </b>
            </center>
      </body>
</html>
```

JSP implicit objects
==========
 To make the coding easy in jsp there are few object which are readily available to programmers.
 There are 9 jsp objects available

1. request -------> HttpServletRequest(I)
2. response------> HttpServletResponse(I)
3. config ---------> ServletConfig(I)
4. application---> ServletContext(I)
5. session -------> HttpSession(I)
6. out-------------> JspWriter(AC)
7. page-----------> Object(CC)
8. pageContext---> PageContext(AC)
9. exception-------> Throwable(CC)

request,response
==========
 These object are directly available inside jsp, where we can call the methods associated with HttpServletReqeust,
 HttpServletResponse.

eg:

```
<%@ page language ="java" isELIgnored='false'%>
<h1>Working with Implicit object(9)...</h1>
<h1>
      Request method type is  :: <%= request.getMethod()%><br/>
      Request parameter is      :: <%= request.getParameter("username") %><br/>
      Client ip address is           :: <%= request.getRemoteAddr() %><br/>
      Content type info is         :: <%= response.getContentType() %>
</h1>
```

input
=====
http://localhost:9999/FirstApp/index.jsp?username=sachin

output
======
Working with Implicit object(9)...
Request method type is :: GET
Request parameter is :: sachin
Client ip address is :: 0:0:0:0:0:0:0:1
Content type info is :: text/html

application
=========
      The data should be configured in "ServletContext" object.
      It can be configured only in XML approach.

web.xml

```
-----------
<web-app>
      <display-name>JSP IMPLICIT OBJECT</display-name>
      <context-param>
            <param-name>username</param-name>
            <param-value>iNeuron</param-value>
      </context-param>
</web-app>

index.jsp
------------
<%@ page language ="java" isELIgnored='false'%>
<h1>Working with Implicit object(9)...</h1>
<h1>
      The context parameter UserName is ::
      <%=application.getInitParameter("username")%><br/><br/>
      The application name is :: <%= application.getServletContextName()%>
</h1>

output
---------
Working with Implicit object(9)...
The context parameter UserName is :: iNeuron
The application name is :: JSP IMPLICIT OBJECT

session
======
  It is of type HttpSession,by default available to every jsp page.
  HttpSession methods can be called using "session" object.

index.jsp
------------
<%@ page language ="java" isELIgnored='false'%>
<h1>Working with Implicit object(9)...</h1>
<h1>
      Session id is :: <%= session.getId() %><br/>
      Is Session newly Created :: <%= session.isNew() %> <br/>
      Session Time out is :: <%= session.getMaxInactiveInterval() %> seconds<br/>
</h1>

output
=====
Working with Implicit object(9)...
Session id is :: 0D4130EFB9D3705C8A978998DF79AD1A
Is Session newly Created :: false
Session Time out is :: 1800 seconds


config
--------
   It is of type ServletConfig.
   methods applied on config object
      a. getServletName()
      b. getInitParameter(String name)
      c. getInitParameterNames()
      d. getServletContext()

web.xml
-----------
```

```
<web-app>
      <display-name>JSP IMPLICIT OBJECT</display-name>
      <servlet>
            <servlet-name>DemoJsp</servlet-name>
            <jsp-file>/config.jsp</jsp-file>
            <init-param>
                  <param-name>username</param-name>
                  <param-value>iNeuron</param-value>
            </init-param>
      </servlet>
      <servlet-mapping>
            <servlet-name>DemoJsp</servlet-name>
            <url-pattern>/test</url-pattern>
      </servlet-mapping>
</web-app>

config.jsp
========
<%@ page language ="java" isELIgnored='false'%>
<h1>Working with Implicit object(9)...</h1>
<h1>
      The logical name of the servlet is :: <%= config.getServletName()%><br/><br/>
      The initialization parameter is :: <%= config.getInitParameter("username")%>
</h1>

Scene1:
request
======
http://localhost:9999/FirstApp/config.jsp

response
========
Working with Implicit object(9)...
The logical name of the servlet is :: jsp
The initialization parameter is :: null

Scene2:
request
---------
http://localhost:9999/FirstApp/test

response
------------
Working with Implicit object(9)...
The logical name of the servlet is :: DemoJsp
The initialization parameter is :: iNeuron

note: To reflect servlet level web.xml file configuration in jsp ,compulsorily we
should access it through 'url-pattern'.

pageContext
===========
 It is a implicit object of type PageContext.
 using pageContext object we can perform 3 things
      a.  We can get all the other implicit objects not in jsp ,but in "custom
action tags".
      b.  To peform attribute management in scope(Jsp scopes).
      c.  To perform request dispatching mechanism we use pageContext.
```

RequestDispatching mechanism in jsp
===========================
	public void forward(String target)
	public void include(String target)
		note: Target resource can be supplied either using relative path or absolute path.

first.jsp

```
<h1>This is First JSP</h1>
<%
	pageContext.include("second.jsp");
%>
```

second.jsp

```
<h1>Hello This is Second JSP page...</h1>
```

output
======
request
----
http://localhost:9999/SecondApp/First.jsp

response

This is First JSP
Hello This is Second JSP page...

page implicit object
================
1. This implicit object always points to Current Servlet object
	Object page = this;
2. Since page is of type Object, can we make a call to Servlet specific methods?
	Answer. Not possible , we can make a call only to Object class methods.
		If we want to make a call then we need to do explicit TypeCasting.

Which of the following statements are valid?
1. <%= page.getServletInfo() %>//invalid
2. <%= this.getServletInfo() %>//valid
3. <%= getServletInfo() %>//valid
4. <%= ((HttpServlet)page).getServletInfo() %>//valid

note: since on page object we can't make a call to servlet methods, so page object is rarely used object.

2. include directive
		If several jsp pages contains the same code, then it is recomended to seperate that common code in a seperate file.
		Wherever that common code is requried we can "include" that file.
		This mechanism is only called as "include" mechanism, and commonly used nature of include is to add
		"header" and "footer" information which is common in every jsp pages.

Advantages
=========
	a. It promotes code Reusability.

b. It improves the maintainence of the code.
c. Enhancements will become easy.

include directive
==========

```
<%@ include file='' %>
```