Explore More

Subcription : Premium CDAC NOTES & MATERIAL @99

Contact to Join
Premium Group

Click to Join
Telegram Group

For More E-Notes

Join Our Community to stay Updated

TAP ON THE ICONS TO JOIN!

	codewitharrays.in freelance project available to buy contact on 8007592194	
SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

	freelance_Project available to buy contact on 8007592194	
21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+NextJs+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql

41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48	Train Ticket Booking Project	React+Springboot+MySql
49	Quizz Application Project	JSP+Springboot+MySql
50	Hotel Room Booking Project	React+Springboot+MySql
51	Online Crime Reporting Portal Project	React+Springboot+MySql
52	Online Child Adoption Portal Project	React+Springboot+MySql
53	online Pizza Delivery System Project	React+Springboot+MySql
54	Online Social Complaint Portal Project	React+Springboot+MySql
55	Electric Vehical management system Project	React+Springboot+MySql
56	Online mess / Tiffin management System Project	React+Springboot+MySql
57		React+Springboot+MySql
58		React+Springboot+MySql
59		React+Springboot+MySql
60		React+Springboot+MySql

Spring Boot + React JS + MySQL Project List

Sr.No	Project Name	YouTube Link
1	Online E-Learning Hub Platform Project	https://youtu.be/KMjyBaWmgzg?si=YckHuNzs7eC84-IW
2	PG Mate / Room sharing/Flat sharing	https://youtu.be/4P9clHg3wvk?si=4uEsi0962CG6Xodp
3	Tour and Travel System Project Version 1.0	https://youtu.be/-UHOBywHaP8?si=KHHfE_A0uv725f12
4	Marriage Hall Booking	https://youtu.be/VXz0kZQi5to?si=llOS-QG3TpAFP5k7
5	Ecommerce Shopping project	https://youtu.be/vJ_C6LkhrZ0?si=YhcBylSErvdn7paq
6	Bike Rental System Project	https://youtu.be/FIzsAmIBCbk?si=7ujQTJqEgkQ8ju2H
7	Multi-Restaurant management system	https://youtu.be/pvV-pM2Jf3s?si=PgvnT-yFc8ktrDxB
8	Hospital management system Project	https://youtu.be/IynIouBZvY4?si=CXzQs3BsRkjKhZCw
9	Municipal Corporation system Project	https://youtu.be/cVMx9NVyI4I?si=qX0oQt-GT-LR_5jF
10	Tour and Travel System Project version 2.0	https://youtu.be/_4u0mB9mHXE?si=gDiAhKBowi2gNUKZ

Sr.No	Project Name	YouTube Link
11	Tour and Travel System Project version 3.0	https://youtu.be/Dm7nOdpasWg?si=P_Lh2gcOFhlyudug
12	Gym Management system Project	https://youtu.be/J8_7Zrkg7ag?si=LcxV51ynfUB7OptX
13	Online Driving License system Project	https://youtu.be/3yRzsMs8TLE?si=JRI_z4FDx4Gmt7fn
14	Online Flight Booking system Project	https://youtu.be/m755rOwdk8U?si=HURvAY2VnizIyJlh
15	Employee management system project	https://youtu.be/lD1iE3W_GRw?si=Y_jv1xV_BljhrD0H
16	Online student school or college portal	https://youtu.be/4A25aEKfei0?si=RoVgZtxMk9TPdQvD
17	Online movie booking system project	https://youtu.be/Lfjv_U74SC4?si=fiDvrhhrjb4KSlSm
18	Online Pizza Delivery system project	https://youtu.be/Tp3izreZ458?si=8eWAOzA8SVdNwlyM
19	Online Crime Reporting system Project	https://youtu.be/0UlzReSk9tQ?si=6vN0e70TVY1GOwPO
20	Online Children Adoption Project	https://youtu.be/3T5HC2HKyT4?si=bntP78niYH802I7N

JSP Interview Questions

Q. What is JSP Implicit Object?

- **request**: This is the HttpServletRequest object associated with the request.

Example: index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
  <title>Implicit Objects</title>
</head>
<body>
  <form action="request.jsp">
    <input type="text" name="username">
    <input type="submit" value="submit">
  </form>
</body>
</html>
```

request.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
  <title>Implicit Objects</title>
</head>
<body>
  <%  String username = request.getParameter("username");
      out.println("Welcome "+ username);
   %>
</body>
</html>
```

- **response**: This is the HttpServletResponse object associated with the response to the client.

Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
    <title>Implicit Objects</title>
  </head>
<body>
    <%response.setContentType("text/html"); %>
</body>
</html>
```

- **session**: This is the HttpSession object associated with the request.

Example: index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
        <title>Implicit Objects</title>
    </head>
<body>
    <% session.setAttribute("user","Pradeep"); %>
    <a href="session.jsp">Click here to get user name</a>
</body>
</html>
```

session.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
        <title>Implicit Objects</title>
    </head>
<body>
    <% String name = (String)session.getAttribute("user");
        out.println("User Name is " +name);
    %>
</body>
</html>
```

- **out**: This is the PrintWriter object used to send output to the client.

Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
        <title>Implicit Objects</title>
    </head>
<body>
    <% int num1=10;int num2=20;
        out.println("num1 is " +num1);
        out.println("num2 is "+num2);
    %>
</body>
</html>
```

- **application**: This is the ServletContext object associated with the application context.

Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
        <title>Implicit Objects</title>
    </head>
<body>
    <% application.getContextPath(); %>
</body>
</html>
```

- **config**: This is the ServletConfig object associated with the page.

Example: web.xml

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">
    <servlet>
        <servlet-name>comingsoon</servlet-name>
        <servlet-class>mysite.server.ComingSoonServlet</servlet-class>
    </servlet>
```

```
    <servlet-mapping>
        <servlet-name>comingsoon</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>
</web-app>
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
        <title>Implicit Objects</title>
    </head>
<body>
    <% String servletName = config.getServletName();
        out.println("Servlet Name is " +servletName);%>
</body>
</html>
```

- **pageContext**: It is used to get, set and remove the attributes from a particular scope.
- **page**: Page implicit variable holds the currently executed servlet object for the corresponding jsp. Acts as this object for current jsp page.

Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
        <title>Implicit Objects</title>
    </head>
<body>
    <% String pageName = page.toString();
        out.println("Page Name is " +pageName);
    %>
</body>
</html>
```

- **Exception**: It is used for exception handling in JSP.

Example:

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isErrorPage="true"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
        <title>Implicit Objects</title>
    </head>
<body>
    <%  int[] num1={1,2,3,4};
        out.println(num1[5]);
    %>
    <%= exception %>
</body>
</html>
```

Q. How JSP pages are processed, from the request to the server to the response to the user?

When the user `page.jsp` follows the link to the page , he sends an http request to the server `GET /page.jsp`. Then, based on this request and the text of the page itself, the server generates a java class, compiles it and executes the resulting servlet, which forms a response to the user in the form of a representation of this page, which the server redirects back to the user.

Q. What are the phases of the JSP life cycle?

The JSP life cycle consists of several phases that are managed by the JSP container:

- **Translation** - checking and parsing the JSP page code to create the servlet code.
- **Compilation** - compilation of the servlet source code.
- **Class Loading** - loading a compiled class into memory.
- **Instantiation** - the implementation of the constructor without the parameter of the loaded class for initialization in memory.
- **Initialization** - calling the `init()` method of the JSP class object and initializing the servlet configuration with the initial parameters that are specified in the deployment descriptor (`web.xml`). After this phase, the JSP is able to handle client requests. Typically, these phases occur after the first client request (i.e., lazy loading), but you can configure the loading and initialization of JSP at the start of the application, similar to servlets.
- **Request Processing** - the long life cycle of processing client requests with a JSP page. Processing is multithreaded and similar to servlets - for each request a new thread, objects are created, `ServletRequest` and the `ServletResponseservice` methods are executed.

- **Destroy** is the last phase of the JSP life cycle in which its class is removed from memory. This usually happens when you turn off the server or unload the application.

Q. What are the JSP life cycle methods?

A servlet container (for example, Tomcat, GlassFish) creates a servlet class from a JSP page that inherits interface properties `javax.servlet.jsp.HttpJspBase` and includes the following methods:

- **jspInit()**- the method is declared in the JSP page and is implemented using the container. This method is called once in the JSP life cycle in order to initialize the configuration parameters specified in the deployment descriptor. You can override this method by defining a JSP scripting element and specifying the necessary parameters for initialization;
- **_jspService()**- the method is automatically overridden by the container and corresponds directly to the JSP code described on the page. This method is defined in the interface `HttpJspPage`, its name begins with an underscore, and it differs from other life cycle methods in that it cannot be redefined;
- **jspDestroy()**- the method is called by the container to remove the object from memory (at the last phase of the JSP life cycle is Destroy). The method is called only once and is available for redefinition, providing the ability to free resources that were created in `jspInit()`.

Q. How can I prevent direct access to the JSP page from a browser?

There is no direct access to the directory `/WEB-INF/` from the web application. Therefore, JSP pages can be located inside this folder and thereby restrict access to the page from the browser. However, by analogy with the description of servlets, it will be necessary to configure the deployment descriptor:

```
<servlet>
    <servlet-name> Example </servlet-name>
    <jsp-file> /WEB-INF/example.jsp </jsp-file>
    <init-param>
        <param-name> exampleParameter </param-name>
        <param-value> parameterValue </param-value>
    </init-param>
</servlet>

<servlet-mapping>
    <servlet-name> Example </servlet-name>
    <url-pattern> /example.jsp </url-pattern>
</servlet-mapping>
```

Q. What are the main types of JSP tags?

- **JSP expression**: `<%= expression %>`- expression that will be processed with redirecting the result to the output;

- **JSP Scriptlet**: `<% code %>`- The code to add to the method service().
- **JSP declaration**: `<%! code %>`- code added to the servlet class body outside the method service().
- **JSP page**`<%@ page att="value" %>` directive: - directives for the servlet container with parameter information.
- **JSP directive include**: `<%@ include file="url" %>`- a file on the local system that is included when translating the JSP into the servlet.
- **JSP Comment**: `<%-- comment --%>`- Comment; ignored when translating a JSP page to a servlet.

Q. What are the JSP action tags and JSP Action Elements?

Action tag and **JSP Action Elements** provide methods for working with Java Beans, connecting resources, forwarding queries and creating dynamic XML elements. Such elements always begin with recording jsp:and are used directly inside the JSP page without the need for third-party libraries or additional settings.

The most commonly used JSP Action Elements are:

- jsp:useBean
- jsp:include
- jsp:setProperty
- jsp:getProperty
- jsp:forward
- jsp:plugin
- jsp:attribute
- jsp:body
- jsp:text
- jsp:param
- jsp:attribute
- jsp:output

1. jsp:useBean

This action name is used when we want to use beans in the JSP page. With this tag, we can easily invoke a bean.

```
<jsp:useBean id="" class="" />
```

Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
```

```
 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
 <title>Action JSP1</title>
 </head>
<body>
    <jsp:useBean id="name" class="demotest.DemoClass">
</body>
</html>
```

2. jsp:include

It also used to insert a jsp file into another file, just like include directive. It is added during request processing phase

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

3. jsp:setProperty

This property is used to set the property of the bean. We need to define a bean before setting the property

```
<jsp:setproperty name="" property="">
```

4. jsp:getProperty

This property is used to get the property of the bean. It converts into a string and finally inserts into the output.

```
<jsp:getAttribute name="" property="">
```

5. jsp:forward

It is used to forward the request to another jsp or any static page. Here the request can be forwarded with no parameters or with parameters.

```
<jsp:forward page="value">
```

6. jsp:plugin

It is used to introduce Java components into jsp, i.e., the java components can be either an applet or bean. It detects the browser and adds or tags into the file

```
<jsp:plugin type="applet/bean" code="objectcode"
codebase="objectcodebase">
```

7. jsp:param

This is child object of the plugin object described above. It must contain one or more actions to provide additional parameters.

```
<jsp:params>
    <jsp:param name="val" value="val"/>
</jsp:params>
```

8. jsp:body

This tag is used to define the XML dynamically i.e., the elements can generate during request time than compilation time. It actually defines the XML, which is generated dynamically element body.

```
<jsp:body></jsp:body>
```

9. jsp:attribute

This tag is used to define the XML dynamically i.e. the elements can be generated during request time than compilation time It actually defines the attribute of XML which will be generated dynamically.

```
<jsp:attribute></jsp:attribute>
```

10. jsp:text

It is used to template text in JSP pages. Its body does not contain any other elements, and it contains only text and EL expressions.

```
<jsp:text>template text</jsp:text>
```

11. jsp:output

It specifies the XML declaration or the DOCTYPE declaration of jsp. The XML declaration and DOCTYPE are declared by the output

```
<jsp:output doctype-root-element="" doctype-system="">
```

Q. JSP - Servlet - JSP interaction?

"JSP - servlet - JSP" architecture for building applications is called MVC (Model / View / Controller) :

- Model - data classes and business logic;
- View - JSP pages;
- Controller - servlets.

Q. What do you know about PageContext and what are the benefits of using it?

Implicit JSP object - an instance of a class `javax.servlet.jsp.PageContext` provides access to all namespaces associated with a JSP page, as well as its various attributes. The remaining implicit objects are added to `pageContext` automatically.

A class `PageContext` is an abstract class, and its instance can be obtained through a method call `JspFactory.getPageContext()`, and released through a method call `JspFactory.releasePageContext()`.

PageContext has the following set of features and capabilities:

- a single API for serving namespaces of various scopes;

- several convenient APIs for accessing various `public` objects;
- `JspWriter` output mechanism for output;
- page session use service mechanism;
- mechanism for exposing ("showing") the attributes of the directive to `page` the scripting environment;
- mechanisms for sending or including the current request into other application components;
- mechanism for handling exception processes on error page errorpage;

Q. What do you know about the JSP Expression Language (EL)?

JSP Expression Language (EL) is a scripted expression language that allows you to access Java components (JavaBeans) from JSP. Starting with JSP 2.0, it is used inside JSP tags to separate Java code from the JSP to provide easy access to Java components, while reducing the amount of Java code in JSP pages, or even completely eliminating it.

The development of EL was aimed at making it easier for designers who have minimal knowledge of the Java programming language. Before the advent of the expression language, JSP had several special tags such as scriptlets (English), expressions, etc. that allowed you to write Java code directly on the page. Using an expression language, a web designer only needs to know how to organize the call of the corresponding java methods.

The JSP 2.0 expression language includes:

- Create and modify variables.
- Program flow control: branching, performing various types of iterations, etc.
- Simplified access to embedded JSPs.
- Ability to create your own functions.

An expression language is used inside a construct `${ ... }`. A similar construction can be placed either separately or on the right side of the tag attribute setting expression.

Q. How is error handling using JSTL?

JSTL Core Tags `c:catchand` are used to catch and handle exceptions in the service methods of the class `c:if`.

The tag `c:catchcatches` the exception and wraps it in a variable `exception` available for processing in the tag `c:if`

```
<c:catch var ="exception">
   <% int x = 42/0;%>
</c:catch>
<c:if test = "${exception ne null}">
   <p>Exception is : ${exception} <br />
   Exception Message: ${exception.message}</p>
</c:if>
```

Q. How JSP is configured in the deployment descriptor?

To configure various parameters of JSP pages, an element is used jsp-configthat is responsible for:

- management of scriptlet elements on the page;
- control of execution in the language of expressions;
- URL pattern definition for encoding;
- determining the size of the buffer used for objects on the page;
- designation of resource groups corresponding to the URL pattern that should be processed as an XML document.

```
<jsp-config>
    <taglib>
        <taglib-uri> http://company.xyz/jsp/tlds/customtags </ taglib-
uri>
        <taglib-location> /WEB-INF/exampleTag.tld </ taglib-location>
    </ taglib>
</ jsp-config>
```

Q. Is a session object always created on a JSP page, can I disable its creation?

The jsp page, by default, always creates a session. Using a directive pagewith an attribute, sessionyou can change this behavior:

```
<%@ page session ="false" %>
```

Q. What is the difference between JSPWriter and PrintWriter?

`PrintWriter` is the object responsible for recording the contents of the response to the request. `JspWriter` uses an object `PrintWriter` to buffer. When the buffer is full or flushed, it `JspWriter`uses the object `PrintWriter` to write the content in response.

Q. How to disable caching on back button of the browser?

Q. What are the different tags provided in JSTL?

Q. How is JSP better than Servlet technology?

Q. What are the differences between include directive and include action?

Q. Explain the jspDestroy() method.

Q. What is busy spin? Why should you use it?

Q. There are two classes B extends A and C extends B, Can we cast B into C e.g. C = (C) B;

Q. Is ++ operator is thread-safe in Java?

Q. Difference between a = a + b and a += b ?

Q. Can I store a double value in a long variable without casting?

*Q. What will this return 3*0.1 == 0.3? true or false?*

Q. Which one will take more memory, an int or Integer?

Q. Which containers use a border layout as their default layout?

Q. Which containers use a FlowLayout as their default layout?

Q. What are peerless components?

Q. Is there is any difference between a Scrollbar and a ScrollPane?

Q. What is a lightweight and heavyweight component?

https://www.youtube.com/@codewitharrays

https://www.instagram.com/codewitharrays/

https://t.me/codewitharrays Group Link: https://t.me/ccee2025notes

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project