Core Java Eligibility Test : 19-Jan-2018 @01:30 P.M (Friday)

Objective Question : 25 [40 Marks] Subjective Question : 5 [60 Marks]

Portion : Until Core Java - JDBC

Duration Exam : 1 Hour 40 Mins

Basic HTML document

```
<html>
        <head>
        <title> First Page</title>
        </head>
        <body>
        I am creating my First Page
        </body>
</html>
```

Heading Tag

- There are six different heading tags which are available which are used for certain heading to apply they are 1. Chapter Level heading , 2. Page Level Heading , 3. Paragraph Level Heading etc.

Syntax

```
<hn align="left|right|center"> Text </hn>
```

- Here n can be start from 1 to 6.

<center> Tag

- This tag is used to maintain the content in the center.

```
<center> Content </center>
```

Style Tag

- There are different level style tags which are available

<b> or <strong>	: Bold
<i>	: Italic
<u>	: Underline

Horizontal Line

- To implement horizontal line we need to implement the <hr> tag

Syntax

```
<hr width="100%" height="5%" color="color_name"/>
```

<br> Tag

- This tag is used to go to the next line.

- It is a open tag it does not have the end tag.

```
<br/>
```

Demo : Implementing all the Tags

```
<html>
        <head>
        <title> Second Page</title>
        </head>
        <body>
        <h2 align="center"> <u>Resume</u> </h2>
        <hr width="100%" height="5" color="blue"/>
        <h4>Objective</h4>
        <h3> Formatting Tag</h3>
        i. <b>Bold Text </b> <br/>
        ii.<i> Italic Text</i><br/>
        iii.<u>Underline Text</u><br/>
        </body>
</html>
```

List Tags

- There are two different type of list which are available they are

1. Ordered List **2. UnOrdered List**

1. Ordered List

```
<ol type="I| i| 1| A| a">
<li> List Item</li>
</ol>
```

2.UnOrderedList

```
<ul type="disc|circle|square">
</ul>
```

Hyperlink

- To create the link to a web page , mail id , portion of the same page.

- Using anchor tag we can implement hyperlink.

Syntax

```
<a href="path_of_File"> Link Text</a>
```

Demo : Implementing the Tag

```
<html>
        <head>
        <title> Second Page</title>
        </head>
        <body>
        <h2 align="center"> <u>Resume</u> </h2>
        <hr width="100%" height="5" color="blue"/>
        <h3> Hyper Links</h3>
        <a href="http://www.niit.com"> NIIT</a>|&nbsp;&nbsp;&nbsp;&nbsp;
        <a href="http://www.niitstudent.com"> NIITStudent</a>|&nbsp;&nbsp;&nbsp;&nbsp;
        <a href="http://www.gmail.com"> Gmail</a>
        <br/><br/>
        <a href="Page1.html">Page1</a>|
        <a href="C:\Users\Administrator\Desktop\JavascriptDemos\Demo6.html">Javascript
Demo</a>
        <br/><br/>
        <a href="mailto:srinivas.pattnaik@niit.com">Contact Us</a>
        <br/><br/>
        <a href="http://github.com/srinivaspattnaik">
        <img src="github1.png" width="30" height="30"/>
        </a>
        <hr width="100%" height="5" color="blue"/>
        <h4>Objective</h4>
        <h3> Formatting Tag</h3>
        i. <b>Bold Text </b> <br/>
        ii.<i> Italic Text</i><br/>
        iii.<u>Underline Text</u><br/>
        <h3> List Tags Implementation</h3>
        <h4> Ordered List</h4>
        <ol type="A">
                <li> India</li>
                <li> Srilanka</li>
                <li> Pakistan</li>
                <li> Bangladesh</li>
        </ol>
        <h4> UnOrdered List</h4>
        <ul type="square">
                <li> India</li>
                <li> Srilanka</li>
                <li> Pakistan</li>
                <li> Bangladesh</li>
        </ul>
        </body>
</html>
```

Table Tag

- In every website we will implement table tag.

```
<table cellspacing="2" align="left|center|right">
        <tr>
                <td>Row1Column1</td>
                <td>Row1Column2</td>
        </tr>
        <tr>
        <td>Row2Column1</td>
        <td>Row2Column2</td>
    </tr>
</table>
```

Demo : Implementing Table

```
<html>
    <body>
    <table align="center" >
    <tr bgcolor="pink">
        <th>Student ID</th>
        <th>Student Name</th>
        <th>Marks</th>
    </tr>
    <tr>
        <td>S1001</td>
        <td>Sunil</td>
        <td>78</td>
    </tr>
    <tr bgcolor="gray">
        <td>S1002</td>
        <td>Vinod</td>
        <td>67</td>
    </tr>
    <tr>
        <td>S1003</td>
        <td>Tarun</td>
        <td>68</td>
    </tr>
    <tr bgcolor="gray">
        <td colspan="2">Total Marks</td>
```

```
                <td>68</td>
        </tr>
        </table>

        </body>
</html>
```

18/01/18

Html Form Component

- Form component are basically used to send the data to the server.

- To implement the Form we need to use the following syntax

```
<form action="Resource_Server" method="get|post">
//UI component
</form>
```

- The Resource_Server can be anything like a servlet , jsp , asp.net , php etc.

- get and post are the way how we are sending the data from the client to the server.

Get Vs Post

Get - In this kind of method the data will be sent in the address bar. This is not secured as the data will be sent in address bar. It is fast compare to Post.

Post - In this kind of method the data will be sent in encryption mode. The data is secured in this method. It is slow compare to the Get.

Form UI components

i. Text Field

- To create a one line text field we need to use this syntax

```
<input type="text" name="uname" />
```

ii. Button

- There are three kind of buttons which are available they are

```
<input type="submit" value="Submit"/>
<input  type="reset" value="Reset"/>
<input type="button" value="Click"/>
```

iii. CheckBox

- From multiple options if we want to check multiple options then we can use the checkbox.

```
<input type="checkbox" name="hobbies" value="Browsing"/>Browsing
<input type="checkbox" name="hobbies" value="Reading"/>Reading
<input type="checkbox" name="hobbies" value="Swimming"/>Swimming
```

iv. Radio Button

- From multiple options if we want to check one of the option then we can use the checkbox.

```
<input type="radio" name="gender" value="M"/>Male
<input type="radio" name="gender" value="F"/>Female
```

v. Text Area

```
<textarea cols="20" rows="10" name="addr"> </textarea>
```

-Demo : Implementing Form

```
<html>
<body>
<form action="Register" method="post">
        <table align="center" cellspacing="3">
        <tr bgcolor="pink">
        <td colspan="2"> <center>Sign Up Here</center</td>
        </tr>
        <tr>
        <td>User Name</td><td><input type="text" name="uname" required/></td>
        </tr>
        <tr bgcolor="gray">
        <td>Password</td><td><input type="password" name="passwd" required/></td>
        </tr>
        <tr>
        <td>Customer Name</td><td> <input type="text" name="cname" required/></td>
        </tr>
```

```
            <tr>
            <td bgcolor="gray">Gender</td><td> <input type="radio" name="gender" value="M">Male
             <input type="radio" name="gender" value="F">Female</td>
            </tr>
            <tr>
            <td>Country</td><td> <select name="country">
            <option value="India">India</option>
            <option value="Srilanka">Srilanka</option>
            <option value="Pakistan">Pakistan</option>
            </select></td>
            </tr>
            <tr>
            <td bgcolor="gray">
            Addres</td><td> <textarea cols="20" rows="5"></textarea></td>
            </tr>
            <tr>
            <td>Mobile</td><td> <input type="mobile" pattern="[789]\d{9}" name="mobile"
required/></td>
            </tr>
            <tr bgcolor="pink">
            <td colspan="2">
                        <center><input type="submit" value="Sign Up"/></center>
            </td>
            </tr>
            </table>
</form>
</body>
</html>
```

Java Script

- Javascript is a client side scripting language basically used for validation in the client side.

- It has also used for implementing any kind of animation.

- Javascript has developed by Netscape.

- This script can be embedded into the html program.

- To implement the javascript in our html program we need to use the following syntax.

```
<script type="text/javascript" language="javascript">
</script>
```

- Javascript it will be executed in the browser.

- <script> tag can be implemented in <head> or <body> tag.

Built in Methods in Javascript

- There are various built in methods which are available in javascript some of them are

i. alert() **ii. confirm()** **iii. parseInt()** **iv.prompt()**

i. alert()

- It is used for displaying any message.

```
<html>
<body>
<script language="javascript">
alert("This is Javascript Example");
</script>
</body>
</html>
```

ii. prompt()

- To get any value from the user then we can use this prompt() method.

```
<html>
<body>
```

```
<script language="javascript">
var v=prompt("Enter Your Value");
alert("The Entered Value:"+v);
</script>
</body>
</html>
```

- Here v is the variable of type variant which can store any kind of value.

iii. confirm()

- To get the Yes or No type of value from the User then we can use this method.

```
<html>
<body>
<script language="javascript">
var response=confirm("Do You wish to Continue");
if(response==true)
        alert("Continue");
else
        alert("Stop");

</script>
</body>
</html>
```

iv. parseInt()

- parseInt() method is used for converting the variant type of integer equivalent.

```
<html>
<body>
<script language="javascript">
var v1=prompt("Enter First Value");
var v2=prompt("Enter Second Value");
```

```
num1=parseInt(v1);
num2=parseInt(v2);
num3=num1+num2;
alert("Total:"+num3);
</script>
</body>
</html>
```

User Defined Function

- We can write the user defined functions in our javascript.

Syntax

```
        function function_name([parameters])
        {
        //code goes here
        }
```

```
<html>
<body>
<script language="javascript">
function displayAlert()
{
        alert("I am in Function");
}
displayAlert();
</script>
</body>
</html>
```

Passing Parameters to Function

```
<html>
<body>
<script language="javascript">
```

```
function sum(num1,num2) //declaring
{
        v1=parseInt(num1);
        v2=parseInt(num2);
        alert(v1+v2);
}
sum(10,20); //calling
</script>
</body>
</html>
```

19/01/18

User defined functions in <head> Tag

- We can write the user defined function in <head> section also.

Demo : Implementing a simple Userdefine Function

```
<html>
<head>
        <title>User Defined Function</title>
        <script language="javascript">
        function display()
        {
        alert("Display Function-Head Section");
        }
        </script>
</head>
<body>
<script language="javascript">
display(); //calling
```

```
</script>
</body>
</html>
```

Even Handling in Javascript

- In java script we can have event using the html UI component.

- There are various events which we can implement using javascript they are

i.onclick() **ii.onmouseover()** **iii.onkeypress()**

Demo : Implementing event

```
<html>
<head>
        <title>User Defined Function</title>
        <script language="javascript">
        function display()
        {
        alert("Display Function-Head Section-Event");
        }
        </script>
</head>
<body>

<center>
        <input type="button" value="Click" onclick="display()"/>
</center>


</body>
</html>
```

Demo : Implementing the multiple events in Javascript

```
<html>
<head>
        <title>User Defined Function</title>
        <script language="javascript">
        function display()
        {
        alert("Display Function-Click Event");
        }
        function show()
        {
        alert("Show Function-MouseOver Event");
        }
        </script>
</head>
<body>
<center>
        <input type="button" value="Click" onclick="display()"/>
        <input type="button" value="MouseOver" onmouseover="show()"/>
</center>
</body>
</html>
```

External Javascript File

- We can create external javascript file with an extension of .js.
- We can write all the code related to the javascript and we can import that file into the html file.

MyScript.js

```
        function display()
        {
        alert("Display Function-Click Event");
        }
        function show()
        {
        alert("Show Function-MouseOver Event");
        }
```

ScriptDemo.html

```
<html>
<head>
        <title>User Defined Function</title>
        <script language="javascript" src="MyScript.js"></script>
</head>
<body>
<center>
        <input type="button" value="Click" onclick="display()"/>
        <input type="button" value="MouseOver" onmouseover="show()"/>
</center>
</body>
</html>
```

CSS - Cascading Style Sheet

- Stylesheet is basically used to have a common look and feel to every component or for the entire website.

- CSS also will make the general html component to look rich.

- CSS can be divided into three different parts based on their usage.

1. Inline StyleSheet

2. Embedded Stylesheet

3. External Stylesheet

1. Inline Stylesheet

- Basically we can implement the style directly to the html tag.

- This way of implementation will not be able to reuse the style.

Example

```
<p align="justify" style="color:red;font-family:verdana;font-size:5">

Paragraph Text

</p>
```

Demo : Implementing the Inline Style

```
<html>

<body>

<p align="justify" style="color:red;font-family:verdana;font-size:13">
```

- Using spring framework we can overcome these problems by enabling simple java bean(POJO). Minimize the dependence of the application code on its framework. No lookup operation required in this type of framework.

```
</p>

<p>

Pargraph without Style

</p>

</body>

</html>
```

2. Embedded Style

- Embedded style can be implemented in head section of the page and here we can implement or reuse the style to the total web page components.

```
<style type="text/css">

//different styles

</style>
```

Demo : Implementing the Embedded Style

```
<html>

<head>

        <style type="text/css">

        p

        {
```

```
        color:red;
        font-family:Monotype Corsiva;
        font-size:14
        }

        </style>
</head>
<body>
<p>
        Paragraph -1
</p>
<p>
        Pargraph -2
</p>
</body>
</html>
```

<u>Style Blocks</u>

- There are two major style blocks which we can create and use them to the elements of html.

1. id **2.class**

<u>i.id</u>

- This style block can be created with preceding the # symbol.
- Basically the id selector we can use for the unique element.

<u>Syntax</u>

```
#format1
{
        //styles
}
```

Demo : Implementing ID type of selector

```
<html>

<head>

        <style type="text/css">

        #format1

        {

        color:red;

        font-family:Monotype Corsiva;

        font-size:14

        }

        #format2

        {

        color:blue;

        font-family:verdana;

        font-size:15

        }

        </style>

</head>

<body>

<p id="format1">

        Paragraph -1

</p>

<p id="format2">

        Pargraph -2

</p>

</body>

</html>
```

ii. class Selector

- A class type of selector is used when we want a multiple elements want to have the same kind of format then we can use the class type of selector.

- A class type of selector preceded with "." symbol.

Demo : Implementing the class type of Selector

```
<html>
<head>
        <style type="text/css">
        .format1
        {
        color:red;
        font-family:Monotype Corsiva;
        font-size:14
        }
        </style>
</head>
<body>
<p class="format1">
        Paragraph -1
</p>
<p class="format1">
        Pargraph -2
</p>
<h3 class="format1">
        Heading 3
</h3>
</body>
</html>
```

3. External Stylesheet

- External Stylesheet is used when we want to implement a same style to multiple web pages then we can implement.

- This stylesheet extension will be .css.

MyCSS.css

```
        .format1
        {
        color:red;
        font-family:Monotype Corsiva;
        font-size:14
        }
```

CSSDemo4.html

```
<html>
<head>
        <link rel="stylesheet" type="text/css" href="MyCSS.css"/>
</head>
<body>
<p class="format1">
        Paragraph -1
</p>
<p class="format1">
        Pargraph -2
</p>
<h3 class="format1">
        Heading 3
</h3>
</body>
</html>
```

20/01/18

Servlet

Static Web Page

- A static web page is the one which does not have any dynamic activity at the runtime and this is basically used only for design purpose.

- Static web pages we create generally by using HTML and CSS.

Dynamic Web Page :

- A dynamic web page is the one which will do the dynamic activity and here the dynamic activity like

1. Accessing the data from the Database.

2. Accessing any kind of services.

- To implement this kind of activity we need to use the dynamic web resources they are

1. Java Technology	**: Servlet , JSP**	**: Glassfish, JBoss, WebLogic**
2. .Net Technology	**: Asp.Net**	**: IIS - Internet Information Server**
3. PHP Technology	**: PHP**	**: WAMP , XAMP**

Web Server

- Web server is the one container which can contain multiple web components.

- These web components may be static or dynamic.

Web Request

- A web request is the one which will be sent from the client to the server.

- A web request which can be sent using http url.

http://www.yahoo.com

- Here http is the protocol which is called web request.

- A web request will be sent using browser.

Web Response

- A web response is the one which will be sent from the web server to the client.

- Mostly the web response is the html format.

Servlet

- A servlet is a server side component which execute in the server and sends the response to the client.

- A servlet is a simple java program and any java class can be called as servlet if the class has implemented Servlet interface of javax.servlet package.

```
public interface Servlet

{

        public void init(ServletConfig servletConfig);

        public void service(ServletRequest request,ServletResponse response)throws
ServletException,IOException;

        public void destroy();

        public String getServletInfo();

        public ServletConfig getServletConfig();

}
```

<u>Life Cycle of Servlet</u>

- Servlet life cycle has three stages they are

<u>1. init()</u>

- In case of init() method the servlet instance is going to be created and it will be loaded into the memory.

<u>2. service()</u>

- service() method is the one where we can implement what servlet has to do.

- This method will take two objects they are ServletRequest and ServletResponse object.

- This method will process the request and sends the response to the client.

<u>3. destroy()</u>

- This method will be called at the last when no clients are accessing the servlet then the destory method will be gets called.

- Here the instance of the servlet will come out of the memory.

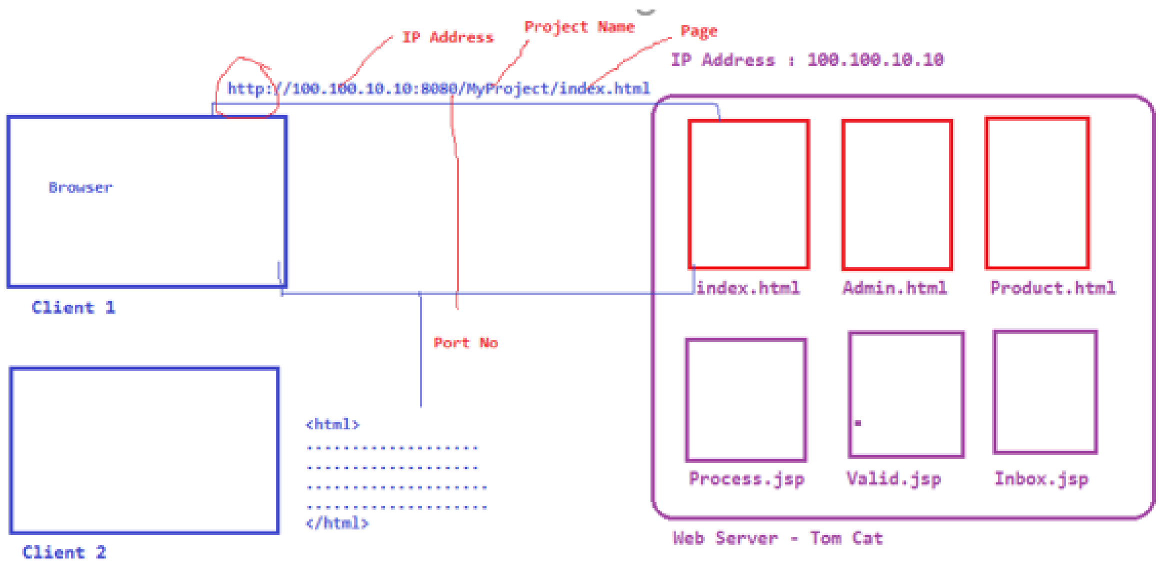
<u>GenericServlet</u>

- GenericServlet is a class which is already implementing the Servlet interface.

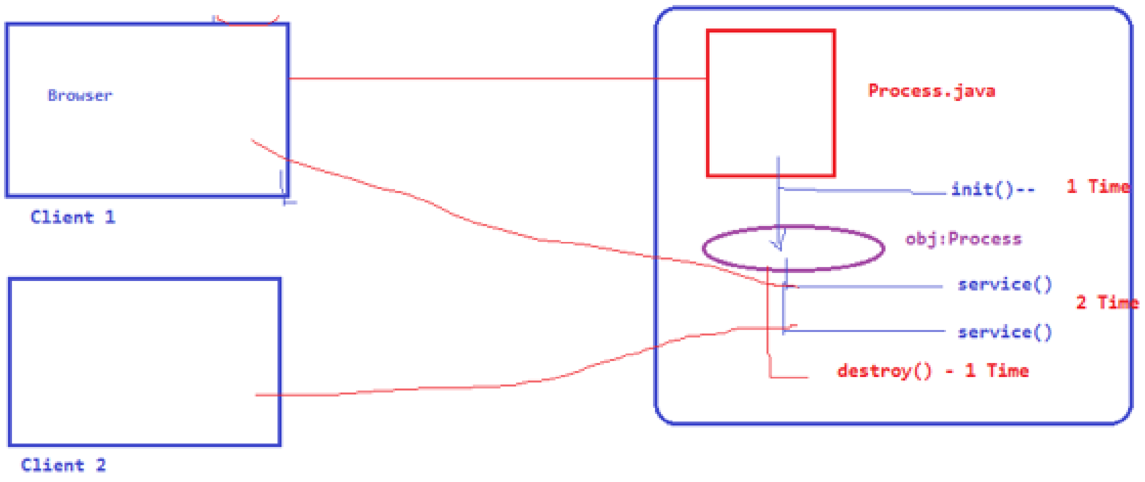
- So we can easily extend the GenericServlet class and write the servlet code.

- Here we need to override only the service() method and we need not to override all the methods.

```
public class MyGenericServlet extends GenericServlet
{
        public void service(ServletRequest request, ServletResponse response)throws
ServletException,IOException
        {
        //code goes here
        }
}
```



Life Cycle of Servlet



22/01/18

Creating a Simple GenericServlet

```
import java.io.*;

import javax.servlet.GenericServlet;

import javax.servlet.*;

import javax.servlet.annotation.WebServlet;


@WebServlet("/MyGenericServlet")

public class MyGenericServlet extends GenericServlet

{

	private static final long serialVersionUID = 1L;


	public void service(ServletRequest request, ServletResponse response) throws
ServletException, IOException

	{

		response.setContentType("text/html");

		PrintWriter out=response.getWriter();


		out.println("<html><body>");

		out.println("<h3><u>Resume</u></h3>");


		out.println("This is the First Servlet");


		out.println("</body></html>");


	}


}
```

- Here service() method which will take the request and response object. This object will be created by the web container.

- response.setContentType("text/html"); - This setContentType() is a method which will basically implement the type of data which response will hold.

- response.getWriter() : This method getWriter() will return a PrintWriter object which will write to the response.

-To call this servlet we need to use the following Url

http://localhost:8083/ServletDemo1/MyGenericServlet

HttpServlet

- HttpServlet is also class which is in javax.servlet.http package.

- HttpServlet has various doXXX() methods which will work as service() methods.

- Here doXXX() methods are nothing but the all the http methods like doGet() , doPost() , doOption() , doDelete() etc.

- get method will be called when the web client calls with get request.

- post method will be called when the web client calls with post request.

When Get and When Post Request will be Sent?

- When the client will call the servlet using the address bar or using hyperlink then doGet() method will be called.

- When the client will call the servlet using form whose method is get then doGet() method will be called.

- When the client will call the servlet using form whose method is post then doPost() method will be called.

Demo : Implementing Get and Post request

index.html

```
<html>

<head>

<title>Insert title here</title>

</head>

<body>

<h3>Calling Servlet using Hyperlink</h3>

<a href="MyHttpServlet">HttpServlet</a>
```

```
<h3>Calling the Servlet using Form whose Method is Get</h3>

<form action="MyHttpServlet" method="get">

<input type="submit" value="HttpServlet"/>

</form>


<h3>Calling the Servlet using Form whose Method is Post</h3>

<form action="MyHttpServlet" method="post">

<input type="submit" value="HttpServlet"/>

</form>


</body>

</html>
```

MyHttpServlet.java

```
import java.io.*;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.*;


@WebServlet("/MyHttpServlet")

public class MyHttpServlet extends HttpServlet

{

	private static final long serialVersionUID = 1L;

	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws 
ServletException, IOException

	{

		response.setContentType("text/html");

		PrintWriter out=response.getWriter();


		out.println("<html><body>");
```

```
                out.println("Get Method");

                out.println("</body></html>");

        }


        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException

        {

                response.setContentType("text/html");

                PrintWriter out=response.getWriter();


                out.println("<html><body>");

                out.println("Post Method");

                out.println("</body></html>");

        }


}
```

- HttpServlet doXXX() method will take two parameters i.e HttpServletRequest and HttpServletResponse and these two interfaces are derived from ServletRequest and ServletResponse.

JSP

- The Full form is Java Server Pages.

- JSP will look like an html page and it will contain the dynamic and static content.

- JSP page have different components. Some of the components are

1. Directive **2.Scriplet** **3.JSP action tags** **4.Built in Objects**

1. Directive

- This component will add the information to the JSP page.

- There are three different directives which are available in JSP they are

1. page **2.taglib** **3.include**

Syntax

<%@taglib_name properties=value%>

1. Page

- This directive is basically used for adding basic information to the JSP page.

- Some of the basic information which we will be added using page directive

1. ContentType	**: Specifies which type of data the client will receive**
2. import	**: Specifies which packages we need to import**
3. session	**: Specifies which whether it will support session**
4. errorPage	**: Specifies which errorPage it has to redirect**
5. isErrorPage	**: Specifies whether the page is an error page.**

2. Taglib

- This directive is basically used for adding any usedefined tag or JSTL tags to our jsp pages.

Syntax

<%@taglib uri="path_uri" prefix="prefix_name"%>

3.include

- This directive is used when we want to add any particular jsp or html page output to our jsp page.

Syntax

<%@include file="jsp|html file name"%>

Scriplets

- Scriplets are basically used to add the java code into our jsp pages.

- There are three different type of scriplets which are available they are

1. Declaration Scriplet **2.Normal Scriplet** **3.Expression Scriplet**

1.Declaration Scriplet

- This scriplet is used when we want to declare certain variables or objects of a class before service() method.

Syntax

```
<%!
        //code goes here
%>
```

2. Normal Scriplet

- The scriplet which is used to write the code with in the service (_jspService) method.

Syntax

```
<%
        //code goes here
%>
```

3.Expression Scriplet

- The expression scriplet is used to write the code with in out.println() method

Syntax

```
<%=(expression)%>
```

23/01/18

Demo : Implementing Scriplet in JSP Page

ScripletDemo.jsp

```
<%@ page language="java" import="java.sql.*" contentType="text/html"%>
<html>
<head>
<title>Scriplet Page - JSP Implementation</title>
</head>
<body>
<h3>Declaration Scriplet</h3>
<%!
        Connection conn;
        ResultSet resultset;
%>
```

```
<h3>Normal Scriplet</h3>
<%
        java.util.Date today=new java.util.Date();
        out.println("Today :"+today);
%>
<h3>Expression Scriplet</h3>
10 + 20 = <%=(10+20) %>

</body>
</html>
```

include directive

- Basically include directive is used to include the other JSP and html page output.

```
<%@include file="Html|JSP File"%>
```

Demo : Implementing a Common Header to all the JSP Pages

Step 1: Create a Header.jsp Page

Header.jsp

```
<%@ page language="java" contentType="text/html"%>
<html>
<head>
<title>Login Page - JSP Implementation</title>
</head>
<body>
<table width="100%" align="center">
        <tr bgcolor="pink">
                <td>
                        <img src="./images/facebook.jpg" width="50" height="100"/>
                </td>
                <td>
                        <h2><font color="blue" face="Monotype Corsiva">
```

```
                                        Blue Sky Inc.
                                </font></h2>
                        </td>
                </tr>
                <tr>
                        <td colspan="2">
                                <p align="right">
                                <a href="index.jsp">Home</a> &nbsp;|&nbsp;
                                <a href="Login.jsp">Login</a>&nbsp;|&nbsp;
                                <a href="Register.jsp">Register</a>&nbsp;|&nbsp;
                                <a href="AboutUs.jsp">About Us</a>&nbsp;|&nbsp;
                                <a href="ContactUs.jsp">Contact Us</a>&nbsp;|&nbsp;
                                </p>
                        </td>
                </tr>
        </table>
        </body>
        </html>
```

Step 2 : Creating a Main Home Page.

index.jsp

```
<%@ page language="java" contentType="text/html"%>
<html>
<head>
<title>Main Page - JSP Implementation</title>
</head>
<body>

<%@include file="Header.jsp"%>

<table align="center">
```

```
<tr bgcolor="pink"><td colspan="2">Example</td></tr>

<tr bgcolor="cyan">

<td>Script Demo :</td><td><a href="ScripletDemo.jsp">Link</a></td>

</tr>

</table>


</body>

</html>
```

Step 3: Create a Login.jsp page

```
<%@ page language="java" contentType="text/html"%>
```

Login.jsp

```
<html>

<head>

<title>Login Page - JSP Implementation</title>

</head>

<body>

<%@include file="Header.jsp"%>

<form action="Register" method="post">

        <table align="center" cellspacing="3">

        <tr bgcolor="pink">

        <td colspan="2"> <center>Sign Up Here</center></td>

        </tr>

        <tr>

        <td>User Name</td><td><input type="text" name="uname" required/></td>

        </tr>

        <tr bgcolor="gray">

        <td>Password</td><td><input type="password" name="passwd" required/></td>

        </tr>

        <tr>

        <td>Customer Name</td><td> <input type="text" name="cname" required/></td>
        </tr>
        </table>
</form>

</body>
</html>
```

Register.jsp

```
<%@ page language="java" contentType="text/html"%>

<html>
<head>
<title>Login Page - JSP Implementation</title>
</head>
<body>

<%@include file="Header.jsp"%>

<form action="Register" method="post">
        <table align="center" cellspacing="3">
        <tr bgcolor="pink">
        <td colspan="2"> <center>Sign Up Here</center></td>
        </tr>
        <tr>
        <td>User Name</td><td><input type="text" name="uname" required/></td>
        </tr>
        <tr bgcolor="gray">
        <td>Password</td><td><input type="password" name="passwd" required/></td>
        </tr>
        <tr>
        <td>Customer Name</td><td> <input type="text" name="cname" required/></td>
```

```
          </tr>
          <tr>
          <td bgcolor="gray">Gender</td><td> <input type="radio" name="gender" value="M">Male
           <input type="radio" name="gender" value="F">Female</td>
          </tr>
          <tr>
          <td>Country</td><td> <select name="country">
          <option value="India">India</option>
          <option value="Srilanka">Srilanka</option>
          <option value="Pakistan">Pakistan</option>
          </select></td>
          </tr>
          <tr>
          <td bgcolor="gray">
          Addres</td><td> <textarea cols="20" rows="5"></textarea></td>
          </tr>
          <tr>
          <td>Mobile</td><td> <input type="mobile" pattern="[789]\d{9}" name="mobile"
required/></td>
          </tr>
          <tr bgcolor="pink">
          <td colspan="2">
                    <center><input type="submit" value="Sign Up"/></center>
          </td>
          </tr>
          </table>
</form>
</body>
</html>
```

MVC Implementation

- Model View Controller architecture or the design pattern which can be used in different web application.

-MVC is not specific to java technology it can be implemented to the various other technology.

- Basically design patterns are used to make the application component reusable , avoiding complexity and enhancing performance for the application.

- There are three major components which we generally come across in MVC pattern they are

1. Model :

- Model component is used for maintaining the business logic and database connectivity.

- Generally the model component is implemented using POJO - Plain Old Java Object.

2. Controller:

- Controller component is used for implementing the various activities they are

1. Security Checking

2. Logging

3. Transactions

- This component will act as mediator between the JSP page to the model component.

- This component also having responsibility to show the different views.

- Servlet will act as Controller.

3. View :

- This component is basically for the end user and which interacts with enduser.

- Basically html page or jsp page will be implemented as view.

24/01/18

Backend Project

- This project which will be created using Maven Quick start Project.

- The major components which will be the part of this project they are

1. DAO Components 2.Model Components 3.TestCases

4. Configuration File

1. DAO Components - Data Access Object

- This component which will be used for data manipulation. This is a simple java class which will have various method by which we do the database operations.

- These components are as following.

1. ProductDAO **2.CategoryDAO** **3.SupplierDAO**

4. UserDAO **5.CartDAO**

2.Model Component

- The model component is used for storing the data in the database.

- These are the entities whose value will be stored in the Database.

- These are implemented using POJO - Plain Old Java Object

- Some of the POJO classes are

1. Product **2.Category** **3.Supplier** **4.User** **5.Cart**

1. Product	: productId,productName,price,quantity,desc,image,categoryId,supplierId
2. Category	: categoryId,categoryName,desc
3. Supplier	: supplierId, supplierName,addr
4. User	: username,password, customerName,Address,mobileNo,email
5. Cart	: cartId,productId,productname,price,quantity

3. Test Cases

- Test cases are implemented basically to test the DAO components.

- Here we will implement unit test cases by using JUnit implementation.

- All the test cases basically implemented for DAOs.

- Some of the test cases which we will implement they are

1. ProductDAOTestCase **2.CategoryDAOTestCase** **3.SupplierDAOTestCase**

4. UserDAOTestCase **5.CartDAOTestCase**

4.DBConfig.java

- Implementation of configuration of Hibernate.

Frontend

- Frontend can be implemented using the different components some of them are

1. JSP Pages 2.SpringControllers 3.Configuration Files

1. JSP Pages

- These will act as view component to the end users and here we will implement the JSTL and other expression languages.

2.SpringControllers

- These are again a simple java class or POJO- Plain Old Java Object

- These classes will take the data from the jsp page and send to the appropriate DAO.

- This controller also display the view pages to the user after it has get the result from the DAO.

3.Configuration Files

- These configuration files like web.xml and dispatcher-servlet.xml file which are used for configuring.

web.xml -> Dispatcher Servlet and Context Loader and Security configuration information.

dispatcher-servlet.xml - Mapping Info , View Resolver info.

Hibernate

- Hibernate is an ORM (Object Relational Mapping) framework.

- ORM framework is used to map the object or entity of the object oriented system to that of entity of the relational database i.e table.

- Hibernate framework has six different components they are

1. Configuration 2.SessionFactory 3.Session 4.Criteria 5.Query 6.Transaction

1. Configuration

- This object will contain the configuration related information of database.

- This object will be created by using a configuration file called hibernate.cfg.xml.

- Some of the tags which contain in this configuration file they are

1. Driver Class **2.Connection Url** **3.User Name**

4. Password

- We can create this object using the following.

```
Configuration config=new Configuration();
config.configure("hibernate.cfg.xml");
```

hibernate.cfg.xml

```
<?xml version="1.0"?>

<hibernate-configuration>
<session-factory>

   <property name="hibernate.dialect">org.hibernate.dialect.DerbyDialect</property>
   <property name="hibernate.connection.driver_class">org.h2.Driver</property>
   <property name="hibernate.connection.url"></property>
   <property name="hibernate.connection.username">cdtje</property>
   <property name="hibernate.connection.password">cdtje</property>
</session-factory>
</hibernate-configuration>
```

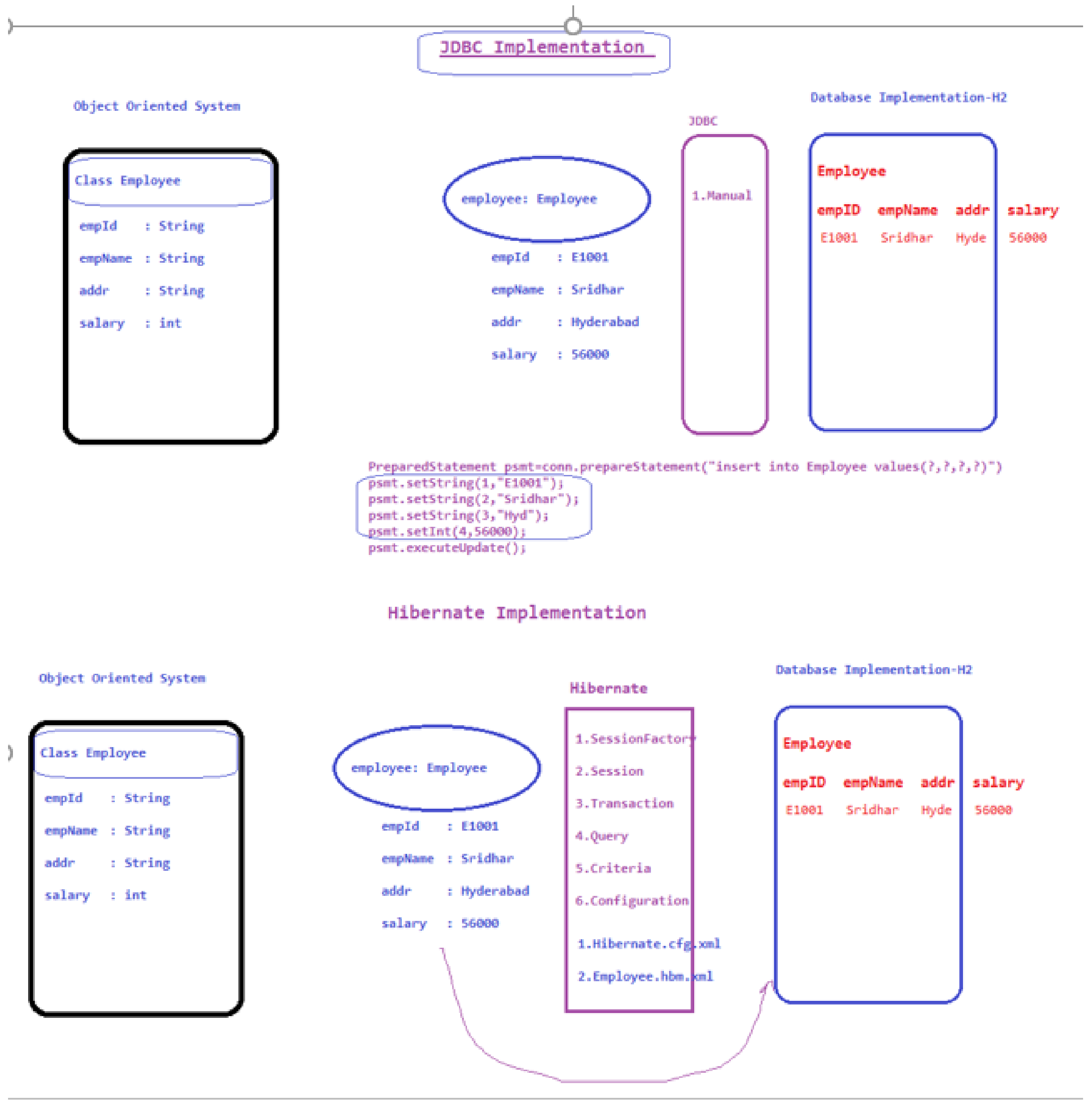
2.SessionFactory

- This object is a kind of factory object which has Session.

- Each session will be used for connectivity with the database.

```
SessionFactory sessionFactory=configure.buildSessionFactory();
```

Hibernate Youtube Videos - Java Brain Hibernate , Gontu Series



```
PreparedStatement psmt=conn.prepareStatement("insert into Employee values(?,?,?,?)")
psmt.setString(1,"E1001");
psmt.setString(2,"Sridhar");
psmt.setString(3,"Hyd");
psmt.setInt(4,56000);
psmt.executeUpdate();
```

25/01/18

3. Session

- This particular component will basically used for doing all the CRUD operation on database.

- Some of the CRUD operations are

1. Insert 2.Delete 3.Update and retrieve the data.

- Session will have all the methods to do the CRUD operations.

- To create the session we have different method they are

1. getCurrrentSession() **2.openSession()**

1. getCurrentSession() : This method will return the current session and it need not be closed.

```
Session session=sessionFactory.getCurrrentSession();
```

2.openSession(): This method will return a new Session object which we need to close explicitely.

```
Session session=sessionFactory.openSession();
```

4. Transaction

- A transaction is a set of operations which need to processed either together or should not be processed.

-Transaction is used to maintain the consistency in the database.

- Syntax

Transaction transaction=session.beginTransaction();

- A transaction need to be committed if everything is successful. A transaction if it is failed then it has to be rolledback.

- transaction.commit() : This is a method to commit the transaction.

- transaction.rollback(): This is a method to rollback the transaction.

- Transaction will have four properties which it should maintain. This property is called ACID properties.

1. Atomicity : In this property either all the transactions need to be implemented and none of them should be implemented.

2. Consistency : In this property after the transaction completes the values should have consistent.

3. Isolation : Each and every transaction should run isolatedly and it should not effect other transaction.

4. Durability : The transaction after it has completed will be stored in the permanent location.

5. Query

- Query is an object by which we can Query the table of the database and fetch it from.

Query query=sessionFactory.createQuery("from Product");

- Once the query is object is created we can get or retrieve the data from it.

- Whenever we implement the Query we use the HQL - Hibernate Query Language.

- query object has different method they are like list() etc which will return resultset.

```
ArrayList<Product> listProducts=query.list();
```

6. Criteria

- This object we will implement when we want to implement criteria based queries.

- This object is used when we want to implement the conditions on query

- Here we can implement criteria.

```
Criteria criteria=session.createCriteria(Product.class);

List myProductList=criteria.list();
```

- To implement the criteria

```
criteria.add(Restrictions.eq("prodname","Moto G"));

criteria.add(Restrictions.gt("price",new Integer(2000));
```

- Here eq is a method of class Restrictions. eq means equal to

Demo : Implementing Hibernate

Step 1: Create a simple Project of java application.

Step 2: Add the Hibernate and H2 dependencies.

Step 3: Create a Model component called Product

Product.java

```
package com;

public class Product

{

		private String prodId;

		private String prodName;

		private int price;
```

```
	public String getProdId()
	{
		return prodId;
	}
	public void setProdId(String prodId)
	{
		this.prodId = prodId;
	}
	public String getProdName()
	{
		return prodName;
	}
	public void setProdName(String prodName)
	{
		this.prodName = prodName;
	}
	public int getPrice()
	{
		return price;
	}
	public void setPrice(int price)
	{
		this.price = price;
	}
}
```

Step 4: Implement the Hibernate Configuration File called hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<hibernate-configuration>
   <session-factory>
      <property name="hbm2ddl.auto">update</property>
      <property name="hibernate.dialect">org.hibernate.dialect.H2Dialect</property>
      <property name="hibernate.driver_class">org.h2.Driver</property>
      <property name="hibernate.connection.url">jdbc:h2:tcp://localhost/~/DT255</property>
      <property name="hibernate.connection.username">dteja</property>
      <property name="hibernate.connection.password">dteja</property>
      <mapping resource="Product.hbm.xml"/>
   </session-factory>
</hibernate-configuration>
```

Step 5: Implement a mapping file called Product.hbm.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<hibernate-mapping>
          <class name="com.Product" table="Product">
             <id name="prodId">
                   <generator class="assigned"></generator>
          </id>
      <property name="prodName"></property>
      <property name="price"></property>
          </class>
</hibernate-mapping>
```

Step 6: Creating a Simple Demo.java

Demo.java

```
import org.hibernate.cfg.Configuration;
import org.hibernate.*;
public class Demo
{
	public static void main(String arg[])
	{
		Configuration config=new Configuration();
		config.configure("hibernate.cfg.xml");
		SessionFactory sessionFactory=config.buildSessionFactory();
		Session session=sessionFactory.openSession();
		Transaction transaction=session.beginTransaction();
		com.Product product=new com.Product();
		product.setProdId("P1002");
		product.setProdName("Samsung Galaxy");
		product.setPrice(11000);
		session.save(product);
		transaction.commit();
		session.close();
		System.out.println("Object Saved");
	}
}
```

29/01/18

Retrieving a Particular Object using load() or get() method

- load() and get() method basically used for getting a particular object from the database table.
- These method return values will be an object of the type for which we have inserted into the table.
- These two methods are in Session object itself.

load() Method

- This method retrieves a particular object from the persistent location i.e database.

- If the object not found in the database then exception will be throw for this method.

Example

```
Product product=(Product)session.load(Product.class,primary_keyvalue);

Product product=(Product)session.load(Product.class,1001);
```

get() Method

- This method retrieves a particular object from the persistent location i.e database

- If the object has not found in the database then null value is returned.

- Once we have retrieved the object using either load() or get() method then we can either update or delete the object.

Example

```
Product product=(Product)session.get(Product.class,primary_keyvalue);
```

HQL - Hibernate Query Language

- Hibernate provides us with its own query language that is HQL.

- We dont require any changes in the query if database changes.

- Hibernate support native sql queries to enable JDBC application.

HQL

- This is query which an object oriented extension of SQL.

- The syntax will look like as sql.

- It automatically populates an object with the data retrieved from the columns.

- It will also support sorting.

Create and Execute Query

- We can create the query using createQuery() method of the session object.

- This method will return a Query interface object which is in org.hibernate.Query package.

Query query=session.createQuery("from Employee");

from=> Which data source whose data is retrieved.

select=> Retrieve specific properties of an object from a table.

Where => It is used to retrieve objects from the database table based on condition.

<u>Demo : Implementing Simple Query</u>

```
import java.util.ArrayList;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import org.hibernate.query.Query;
import com.Product;
public class QueryDemo
{
	public static void main(String arg[])
	{
		Configuration config=new Configuration();
		config.configure("hibernate.cfg.xml");
		SessionFactory sessionFactory=config.buildSessionFactory();
		Session session=sessionFactory.openSession();
		Query query=session.createQuery("from Product");
		ArrayList<com.Product> listProducts=(ArrayList<com.Product>)query.list();
		for(Product product:listProducts)
		{
			System.out.print(product.getProdId()+"\t");
			System.out.print(product.getProdName()+"\t");
			System.out.println(product.getPrice()+"\t");
		}
		session.close();
	}
}
```

Demo : Implementing Parameterized Queries.

```
import java.util.ArrayList;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import org.hibernate.query.Query;

import com.Product;

public class QueryDemo

{

        public static void main(String arg[])

        {

                Configuration config=new Configuration();

                config.configure("hibernate.cfg.xml");

                SessionFactory sessionFactory=config.buildSessionFactory();

                Session session=sessionFactory.openSession();

                Query query=session.createQuery("from Product where prodname=:prodname1");

                query.setParameter("prodname1","Moto G");

                ArrayList<com.Product> listProducts=(ArrayList<com.Product>)query.list();

                for(Product product:listProducts)

                {

                        System.out.print(product.getProdId()+"\t");

                        System.out.print(product.getProdName()+"\t");

                        System.out.println(product.getPrice()+"\t");

                }


                session.close();

        }

}
```

Spring Framework

- The most important part of the web application development is implementing business layer.

Business Layer

- Access the data from Presentation Layer.

- Process the Data

- Store the Data in the database

- Interact with presentation layer and the database layer

- Traditionally many application has implemented business layer using EJB (Enterprise Java Bean).

- Some of the disadvantages of EJB

- Unit Testing is very difficult.

- Application code is very close to the container hence testing is very difficult.

- EJB runs on only Application server and does not run on web server where as Spring framework can run any where.

Spring Framework

- Using springframework we can overcome the problems by enabling POJO.

- Minimizing the dependence of application code on its framework.

- No lookup operation required in this type of framework.

Features

i.Pluggability :

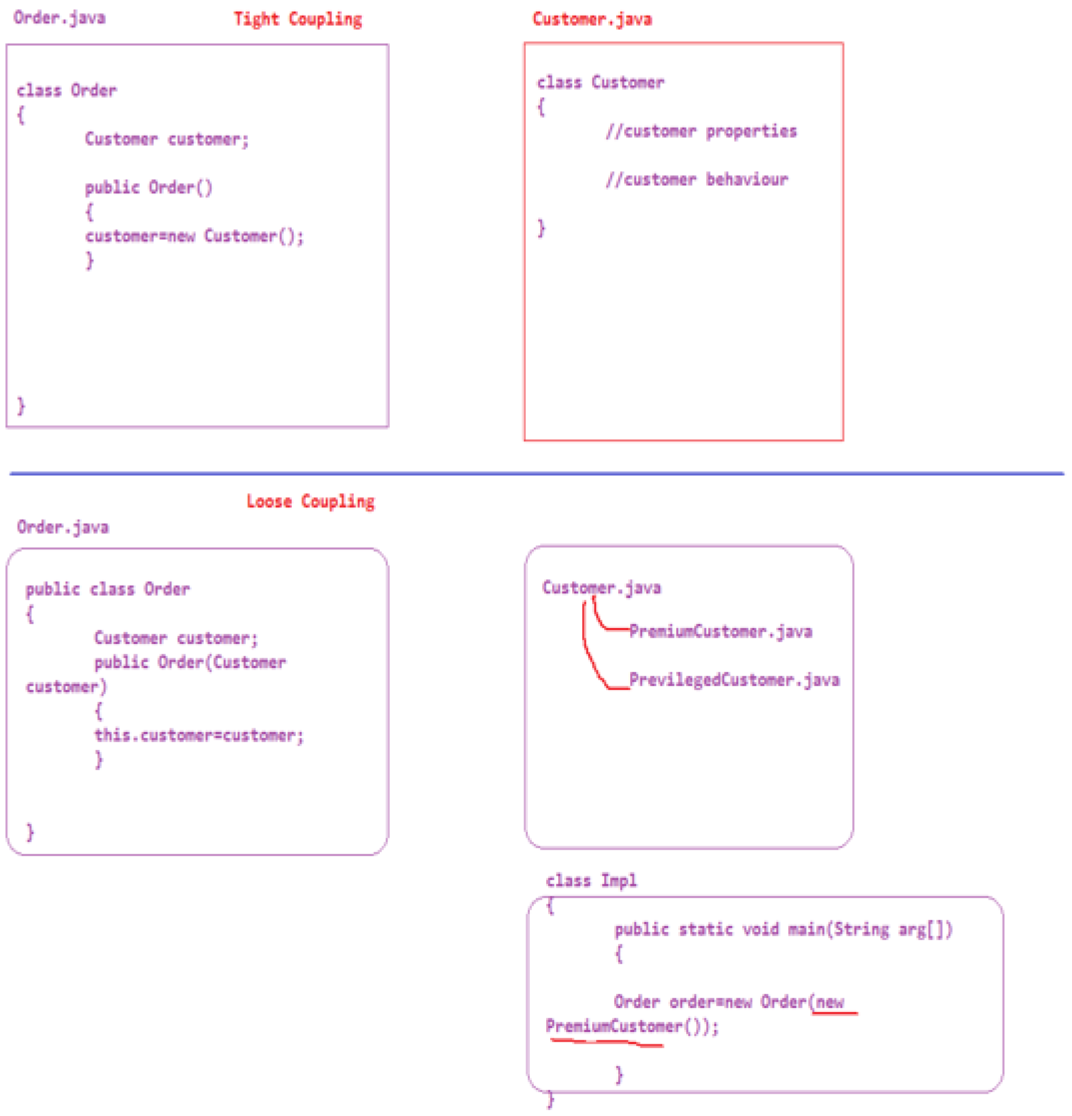
- It allows to associate business layer object with each other using xml configuration file.

- We can plug any new services and modify the existing services.

ii.Dependency Injection

- Spring framework will makes feasible for loose coupling.

- Here a particular object is not dependent fully on other objects.



```
class Order
{
        Customer customer;

        public Order()
        {
        customer=new Customer();
        }
}
```

```
class Customer
{
        //customer properties

        //customer behaviour
}
```

```
public class Order
{
        Customer customer;
        public Order(Customer
customer)
        {
        this.customer=customer;
        }

}
```

```
class Impl
{
        public static void main(String arg[])
        {

        Order order=new Order(new
PremiumCustomer());

        }
}
```

30/01/18

Spring Framework

- The most important part of the web application development is implementing business layer.

Business Layer

- Access the data from Presentation Layer.

- Process the Data

- Store the Data in the database

- Interact with presentation layer and the database layer

- Traditionally many application has implemented business layer using EJB (Enterprise Java Bean).

- Some of the disadvantages of EJB

- Unit Testing is very difficult.

- Application code is very close to the container hence testing is very difficult.

- EJB runs on only Application server and does not run on web server where as Spring framework can run any where.

Spring Framework

- Using springframework we can overcome the problems by enabling POJO.

- Minimizing the dependence of application code on its framework.

- No lookup operation required in this type of framework.

Features

i.Pluggability :

- It allows to associate business layer object with each other using xml configuration file.

- We can plug any new services and modify the existing services.

ii.Dependency Injection

- Spring framework will makes feasible for loose coupling.

- Here a particular object is not dependent fully on other objects.

iii. Aspect Oriented Programming

- Spring framework separates the application logic from the system level services such transaction management , logging and security.

- These system level services are called as aspects which has to run in a separate module.

- These are also called as cross cutting concern.

iv. Spring Container

- This is responsible for managing the life cycle and configuration of all the application objects.

- As a developer we need to need to just create a class and configure them.

v. Light Weight

- Spring framework make it easy to configure and create complex application.

- Spring module provides a platform for writing loosely coupled code.

Spring Architecture

- There around seven layers which are available in spring they are

1. Core Module **2.AOP** **3.ORM**

4. DAO **5.Web Context** **6.ApplicationContext**

7.MVC

1. Core Module

- Provide the fundamental funcitionality of the spring framework.

- This module contain the bean factory container.

- Bean container is responsible for creating the bean as per definition.

- It is in org.springframework.core package

2. DAO Module (Data Access Object)

- Standardize data access work through technologies like hibernate , jdbc etc.

- Handle the exception.

- Enable us to write database code without writing a complex code.

- It is in org.springframework.dao package.

3.ORM Module (Object Relational Mapping)

- Enable developers to integrate spring framework with several other ORM tools like hibernate, JPA , iBatis , JDO.

- It is based on DAO module.

- It is in org.springframework.orm package.

4.MVC Module (Model View Controller)

- Enable to separate the model and application logic from the UI.

- Promote loose coupling.

- It will integrate the different mvc frameworks like struts, tapestry etc.

- It is in org.springframework.web package.

<u>5.ApplicationContext Module</u>

- This is built on top of core module.

- It is is built on concept of bean factory.

- There are variety of services which can be supported (Email , Remoting).

- To implement this we need to use org.springframework.context package.

<u>Spring Container</u>

- Spring container will manage the application objects using DI.

- This will create the object and we have just describle about that object how to be created.

- There are two different type of spring container which are available they are

1. BeanFactory 2.Application Context

<u>1. Bean Factory</u>

- It is a simple container that provides the DI.

- It is in org.springframework.beans.factory.BeanFactory interface.

- From BeanFactory interface we have several implementation which support various levels they are XmlBeanFactory interface.

- XmlBeanFactory loads the bean on the basis of bean definition given in an xml file.

- In the bean configuration each bean defined by using <bean> tag.

- The <bean> tag will have various information they are

1. id 2.name 3.class 4.scope

```
<beans>
        <bean id="prod" class="com.Product"/>
</beans>
```

- To instantiate the bean factory in our application we need to load the configuration file.

- The configuration file can be loaded from different sources

1. File System 2.Classpath 3.ExternalUrl

```
BeanFactory factory=new XmlBeanFactory(new ClassPathResource("SpringConfig.xml"));
```

<u>Note : Here SpringConfig.xml file will contain that bean definitions which bean we want to use in our application.</u>

- To use the Bean Syntax

- We can retrieve the bean instance which is registered we can call getBean() method.

Product product=(Product)factory.getBean("prod");

2.Application Context

- It is a advanced container with several enterprise level functionalities.

- Spring framwork provides the following used implementation for appliction container.

1. FileSystemXmlApplicationContext

2. ClassPathXmlApplicationContext

3. XmlWebApplicationContext

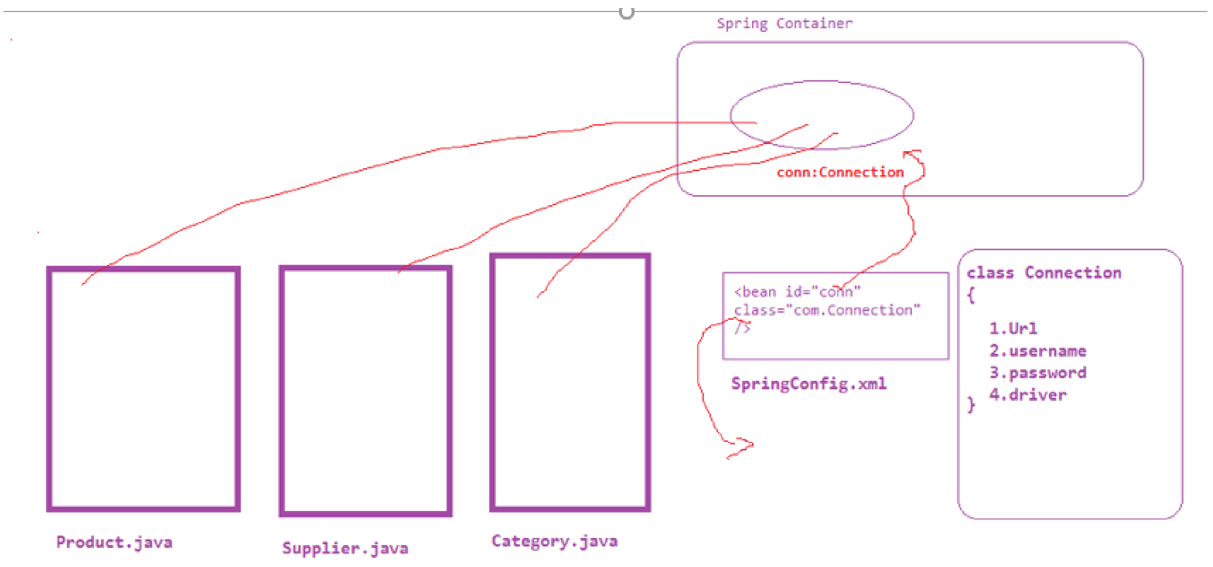
Syntax

ApplicationContext context=new FileSystemXmlApplicationContext("SpringConfig.xml");

BeanFactory Vs ApplicationContext

In the BeanFactory the bean gets created lazily i.e it will be created when we call the getBean() method.

- In the applicationContext the beans does not wait for getBean() method already it will preload.



```
<bean id="conn"
class="com.Connection"
/>
```

```
class Connection
{
  1.Url
  2.username
  3.password
  4.driver
}
```