Servlets (5-12-22 - 27-12-22)

1. **What is Server ?**

▼ **Ans**

- **Server is a software which manages all the resource along with which process the client request and serve the client request.**
  - **3 Types of Servers**
    - ▼ **Database Server**
      - Database server is used to deal with the data.
        - Ex: Oracle, My SQL, MS SQL, Mongo DB, Sybase, Informix, etc.
    - ▼ **Application Server**
      - Application Server is used to execute a Dynamic application or Real time application.
        - Ex: Apache POI, Jetty , JBOSS, IBM WebSphere, Oracle web logic, etc.
        - ▼ **What is Dynamic or Real time application ?**
          - An application which performs all the different types of logic such as Presentation logic, Persistence logic, Business logic is know as a dynamic or real time.
            - Ex: Facebook, Instagram, WhatsApp, etc.
    - ▼ **Web Server**
      - Web Server is used to execute only web applications.
        - Ex: Apache Tomcat, Oracle GlassFish etc.
        - ▼ What is Web Application ?
          - Web application means an application's program that is stored on a remote server and delivered over the Internet through a browser Interface.
          - Ex: Amezon.com

2. **What is Deployment ?**

▼ **Ans**

- **Making all the resources available to the server is know as deployment.**
  - **2 Types of Deployments are there:-**
    - ▼ **Manual Deployment**
      - **In case Manual Deployment, All the resources are made available to the server by manually.**
    - ▼ **Automated Deployment**
      - **In case Automated Deployment, All the resources are made available to the server automatically with the help of automated tools such as ANT, MAVEN, etc.**
      - **By using automated tools like ANT, MAVEN, We can develop Batch File.**
        - ▼ **What is Batch File ?**
          - **A file which can be accessed based on a double click is a Batch File.**

3. **Important Note for the Interview Point : -**

▼ **Ans**

- **Whenever a Web application is compressed, then we deployed it onto the server in the form of war file.**

▼ **What is war file ?**

- war means web archive(compress), it is a compressed version of Web application.

- **All the Web application are deployed onto webapps folder of Apache Tomcat server.**
- **Apache Tomcat server comes in 2 different variants.**
  1. **exe (executable variants)**
  2. **zip (zip file variants)**
- **By defaults, the port number for Apache Tomcat server is 8080.**

4. **Folder of Apache Tomcat Server : -**

▼ **Ans**

- **The different folder of Apache Tomcat server are : -**

▼ **bin**

- It contains a set of startup and shutdown batch file which is used to start and stop the Apache Tomcat server.

▼ **conf**

- It contains a set of configuration with respect to Apache Tomcat server.

▼ **lib**

- It contains set of libraries in the form of Jar file which is used perform some advanced functionalities.

▼ **logs**

- It is used to store all the log messages which are displayed on the server console.

▼ **webapps**

- It is used to deployed all the web application onto Apache Tomcat server.

▼ **work**

- It is used to store the data with respect to translated servlets.

▼ **What is translated servlets ?**

- The conversion of JSP file into servlets is known as translated servlets.
- JSP —> (Java Server Page)

5. Pre-requisites for Apache Tomcat Server (Installation of Apache tomcat Server) ?

▼ Ans

- These are 3 different pre-requisites present for Apache Tomcat server namely

▼ JAVA_HOME

- Open C-Drive, Program Files and Open JAVA and JDK-folder and copy the Entire path which includes all the folder with respect to jdk.

▼ CATALINA_HOME (it is a servlets containers)

- Open the Apache-Tomcat server folder from the respective drive and copy the entire path which include all the older with respect to Apache Tomcat server

▼ PATH

- Select the existing path of system variables and click on edit and click on new and copy paste entire path which includes bin folder of Apache Tomcat server.

▼ JRE_HOME (Optional)

- Open C-Drive, program Files and Open JAVA and JRE folder and copy the entire path which includes all the folder with respect JRE.

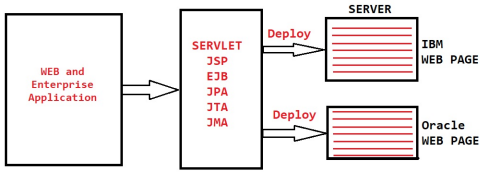
- URL —> http://localhost:8080
- To change port :- Open → conf → server → (open with note pad) → (ctrl+f) → (type “connect”) → (find next for 8times).
- “double click one startup batch file and minimize the server console, Open any Browser and Enter URL=”http://localhost:8080” and hit on Enter”

NOTE: -

1. We cannot run JEE Application on all the servers
2. To run a JEE Application on particular server, the server must mandatorily contains a JEE Container in it.

7. JEE Application ?

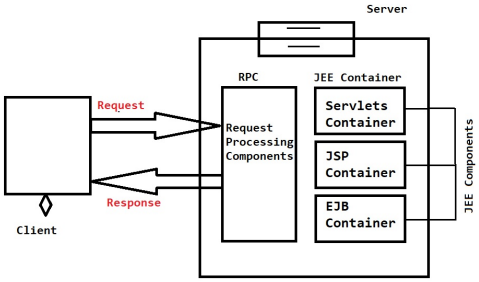
▼ Ans



- **It is a specification for developing both Web and Enterprise Application which is given in the form of abstraction API to achieve loose coupling between Application and Server.**
- Web Resource - [HTML, CSS, JS, JSP, PHP, etc.] (Front End)
- Enterprise Resource - [Servlets, Hibernate, Spring, etc.] (Back End)
- Few of the JEE API are
  1. JSP - Java Server Page.
  2. EJB - Enterprise Java Bean.
  3. JPA - Java Persistence API.
  4. JTA - Java Transaction API.
  5. JMA - Java Mail API.

6. JEE container ?

▼ Ans



- **JEE Container**
  - **It is a engine, which is used to manage all the JEE components such as Servlets, JSP, EJB, etc.**

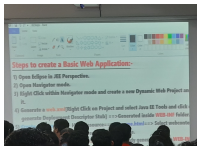
    ▼ Why we call JEE container as an Engine ?

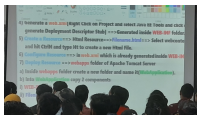
    - Because it performs Multiple tasks such as Application Loading, XML Parsing , Manages Servlet Lifecycle etc.

- **RPC (Request Process Component)**
  - **It is component which take the request and process the request and get back the response.**
- **Server is basically performs two important task namely**
  1. **Manages all the resources.**
  2. **Provides a run-time environment to the JEE Container.**
- **Web-Server is used to execute only Web-Applications.**
- **Application-Server is used to execute both Web and Enterprise Application(JEE Application).**
- With version 6 and above 6. it internally act as Application Server (Apache Tomcat Server)

8. Steps to create a basic Web Project (External way of execution)

▼ Ans





- Click on open perspective and select JEE as a perspective.
- Open Navigation Made
- Right click within the navigator and create a new dynamic web project.
- Generate web.xml (Right click on project and select Java EE tools and click on generate Deployment description stub) web.xml will created inside WEB-INF folder.

▼ **There are different criteria present for any resources in general**

▼ Create Resource

- Create a HTML, Resource with any name which will be generated in web content folder.
- To create HTML resources. (select web content folder → hit control+N → type ht → Name the HTML file)

▼ Configure the Resource

- All the resource must be configured within the web.xml (Deployment descriptor stub)

▼ Deploy the Resource

- All the recourse must be deployed onto the webapps folder of Apache-Tomcat server

- Note:-
  1. Go to Apache-Tomcat → start the server.
  2. Go to any browser and type (http://localhost:8050/basic…..)(folder name which you have created inside webapps)

9. web.xml (or) Deployment description stub

▼ Ans

- **It is a configuration file in xml format which is used to store the information with respect to the configurable resource of an application.**

▼ **What is xml ?**

- **It is a config file which is used to configure the web resources by using custom tag or user defined tags.**
- **To make a resource works efficiently, We use config file.**

- **Each and every application must mandatorily have one web.xml present in it. Without which the JEE container fails to load an application where it throws http 404 error —> Client error—> Recourse not available**

▼ Note

▼ Error code

1. http 404 → Resource not found error
2. http 200 →
3. http 500 →
4. http 100 →

▼ **Difference between XML and HTML ?**

HTML (**Hyper Text Markup Language**)	XML (**eXtensible Markup Language**)
1). We use Inbuilt Tags or Pre-defined tag.	**1). We use Custom Tags or User-defined Tags.**
2). HTML is used to displays data and describes the structure of the webpage. but, It cannot transport the data.	**2). XML stores the data and transfers the data.**
3). HTML is Static	**3). XML is Dynamic.**
4). HTML can ignore small error and It is not case sensitive and also It does not preserve white spaces.	**4). XML does not ignore small error and It is case sensitive and also It can preserve white spaces.**
5). Closing Tags are not necessary.	**5). Closing Tags are necessary.**

▼ UTF-8 (Unicode Transformation Format) 8bit data

- It supports 65,635 character
- It is enhanced ASCII value. All the web.xml is converted to Unicode Transformation Format that is understand by server. (Include chines, , , ,)

▼ ASCII (American Standard Code Information Interchange)

- 255 character

- **By default the root tag for an XML file is <web-app>**
- **The current version of XML is 1.0**
  - <student> —> Root Tag
  - <ID>1</ID> —> Sub Tag
  - <name> Manu </name> —> Sub Tag
  - </student> —> Root Tag

10. How all the configuration made in web.xml is understand by the JEE container ?

▼ Ans

- **All the custom tag or user defined tag present in web.xml are converted into 8-bit Unicode Format. Based on that, All the configuration made in the web.xml are understood by the JEE containers.**
- Servlet = Server Side java Program.

11. Configuration of Application-Tomcat Server in Eclipse (For Internal way of Execution)

▼ Ans

- Before configuration the Apache-Tomcat, Server must not be in use anywhere externally.
- Go To window and select preference.
- Open Server preferences and click on runtime environments.
- Click on add and select the respective version of Apache-Tomcat Server and Click on next.
- Browser for the appropriate version of Apache-Tomcat Server and Open Apache-Tomcat Server folder and click on finish, Apply and close.
- After this just run the Application internally.
- A Server folder will be generated within the navigator.

12. Step for configuration of Apache-Tomcat Server in Eclipse (Internally) ?

▼ Ans

Step 1:- Go to window and select preferences.

Step 2:- Open Server and Select Runtime Environment.

Step 3:- Click on add. Open Apache Folder and Select the respective version of Apache-Tomcat Server used on the system and click on next.

Step 4:- Browser for the appropriate location and open Apache-Tomcat server folder and click on select folder and then click on finish apply and close.

Step 5:- Right click on project and select run as run on server to generate folder inside navigator mode. (Don't delete)

13. Welcome File or Landing Page ?

▼ Ans

- **A file or a page which automatically displays, Whenever the client uses an application is know as Welcome File or Landing page.**
- **index is consider to be a default welcome file or landing page. Since, It is automatically loaded by the JEE container for the first time itself. Whenever the client uses an Application.**
- **We can explicitly make any file as welcome file or landing page by renaming it as index.**

▼ Note:

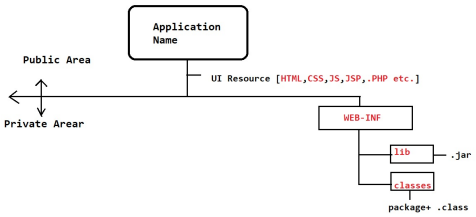
- If configured file are not available then it throws http 404 error (Resource Not Available)

▼ Program

```
<!DOCTYPE html>
<html>
<body bgcolor="yellow">
<h1>UJJWAL IS INNOCENT WHICH IS A JOKE!!</h1>
</bodt>
</html>
```

14. Structure of JEE Application ?

▼ Ans



- **Each and every application must mandatorily have one web.xml present in it without which the JEE container fail to load an application where it throws http 404 error (recourse not available)**
- **We can deploy multiple application onto one single server. But In this case, Each and every application must have a unique name or different name.**
- **Whenever, We start the server all the application are loaded sequentially one after the other by the JEE container**
- **At the time of Application loading the web.xml is passed by the JEE container. Where, If there any error present in web.xml, then the JEE container throws an exception called ParseException.**

15. Static v/s Dynamic Resource ?

▼ Ans

▼ **Static Resource**

- **Resource which are capable of dealing with only static data (which cannot be change) are know as Static Resource.**

  Ex: HTML

▼ **Static Application**

- **An Application which contains only static resources in it is known as a Static Application.**

  Ex: StandAlone Application

  Structure of static application

▼ **Dynamic Resource**

- **Resource which are capable of dealing with only dynamic data (which can be change) are know as Dynamic Resource.**

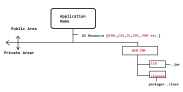
  Ex: Servlet, JSP, PHP, etc.

▼ **Dynamic Application**

- **An Application which contains only dynamic resources in it is known as a Dynamic Application.**

  Ex: Any Real Time Application (Instagram, Facebook, etc.)

  Structure of Dynamic Application



- **Dynamic Application is Always associated with 3different types of Logic namely:**

  ▼ **Presentation Logic**

- **Presentation Logic is used to present the content onto application.**
    - Technologies involved
        1. HTML
        2. CSS
        3. JavaScript
        4. JSP
        5. PHP

▼ **Persistence Logic**

- **Persist means to store. Persistence Logic is used to persist the data into the persistence system.**
    - Technologies involved
        1. JDBC
        2. SQL
        3. Hibernate

▼ **Business Logic**

- **Business Logic is used to perform the core functionalities, that is some set of calculation and validation operation on an application.**
    - Technologies involved
        1. Servlets
        2. Spring

▼ Note

- A Dynamic application or Real Time Application is on Integration of all the 3 different types of logic such as Presentation Logic, Persistence logic and Business Logic.

16. What is Servlets ?

▼ Ans

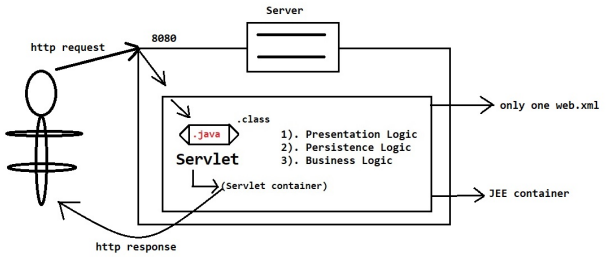
- **Servlet is the server side java program which performs all the 3 different types of Logics. Such as, Presentation Logic, Persistence Logic and Business Logic along with which process the http client request and get back some http response.**

▼ Note 1

- We can deploy n number of application onto one single server. But, In this case, Each and every application must have unique name or different name.
- One single application can contains n number of Servlets in it. But, In this case, Each and every Servlets must have a unique name or different name.

▼ Note 2

- All the application within the server are managed by JEE container.
- All the Servlets within an application are managed by the Servlet container.



- **There are 2 types of Servlets present namely:**
  - ▼ **GenericServlet**
    - Since, It is not specific to any protocol, i.e. independent of protocol. Hence the name called GenericServlet.
      - ▼ protocol
    - GenericServlet does not support "Session". (Session:- Any activity which takes place between the start time and stop time is know as Session.)
    - GenericServlet is an abstract class which is present in javax.servlet package.
    - GenericServlet contains 3 different method in it, Out of one which is an abstract method and the other two are concrete method.
    - The abstract method present in GenericServlet is named as service() method, which has to be overridden for 2 important reasons namely,
      1. service() method is as an abstract method.
      2. Since service() method is only method which takes the parameter called ServletRequest and ServletResponse which is responsible for processing the Client Request.
    - Whenever, We override service() method, It throws an Exception called ServletException and IOException.

```
public void service(ServletRequest req, ServletResponse resp) throws
       ServletException,IOException
```

    - The other 2 methods present in GenericServlet are init() method and destroy() method. Where overriding is optional. Since these are concrete methods.

    ▼ Write a GenericServlet :-

    - Write Servlet class which extends an abstract class called GenericServlest as follows

```
public class FirstServlet extends javax.servlet.GenericServlet
{
  @Override
  public service(ServletRequest req , ServletResponse resp) throw
         ServletException,IOException
    {
        -----------------
        -----------------
    }
}
```

  - ▼ **HttpServlet**
    - Since, It is specific to a particular type of protocol called Http protocol. Hence, the name is HttpServlet
    - HttpServlet supports "Session".
    - HttpServlet is an abstract class present in javax.servlet.http package.
    - HttpServlet contains only concrete methods in it, Without any abstract method.

- In case of HttpServlet, We need to override a respective concrete method called doXXX() for a particular type of Http request.
- Method Signature

```
protected void doXXX(HttpServletRequest req , HttpServletResponse resp)
        throws ServletException,IOException

where:
      XXX --> types of Request
      #   --> protected
      +   --> public
```

- Whenever, We use doXXX(). It throws an exception called ServletException and IOException
- There are 8 different types of Httprequest are present namely
  1. POST
  2. GET
  3. PUT
  4. DELETE
  5. TRACE
  6. OPTION
  7. HEAD
  8. CONNECT

▼ Write a HttpServlet:-

```
public class OurServlet extends javax.Servlet.http.HttpServlet
{
  @Overide
  protected void doXXX(HttpServletRequest req , HttpServletResponse resp)
       throw ServletException,IOException
      {
          //write servlet logic here//
      }
}
```

17. What is Protocol ?

▼ Ans

- Some set rules while transfer to client to server.
- It is a set of rules that need to be followed by the communicating parties in order to have successful data communication.
  - Ex: HTTP (Hyper Text Transfer Protocal)

18. Why class is made as abstract, When it contains only concrete method in it ?

▼ Ans

- So That further Enhancement or modification made, which does not affects the user. (Loose coupling)

20. Is there service() method is present in HttpServlet ?

▼ Ans

- service() method in case of Httpservlet is present as a concrete method which is not a good practice to override. Since HttpServlet depends upon the type of Http request.

```
protected void service(HttpServletRequest req , HttpServletResponse resp)
```

▼ Note

- Web Resource can be accessed based on unique URL pattern.
- Since, Servlet is also Web Resource, It has to be accessed based on unique URL pattern.
- Annotation is supported from JEE 3.x version onwards.
- From JEE 3.x onwards, It is not mandatory to configure a resource in web.xml. Instead Annotation can be used.
- Java Annotation is different and Servlet Annotation is different. (Java Annotation:- Extra information given to a compiler to increase performance of an application)

21. What is difference between a Server and Servlet ?

▼ Ans

- Server is software where as Servlet is Web Resource.

23. In how many ways can we Configure the Resources ?

▼ Ans

1. Using web.xml
2. Using Annotation

22. What are the Criteria for Resources ?

▼ Ans

1. Create Resource
2. Configure Resource
3. Deploy Resource

22. What are the Criteria for Servlet ?

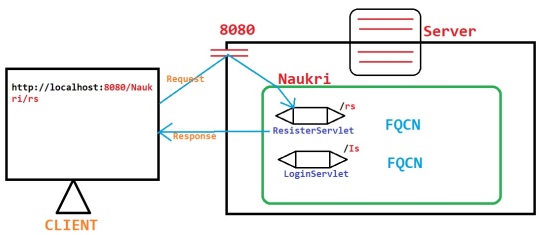
▼ Ans

- Since, Servlet is also Web Resource, There are 3 different criteria present for a Servlet namely:-
  1. Create Servlet
  2. Configure Servlet
  3. Deploy Servlet

24. Configuration of Servlet in web.xml ?

▼ Ans

- Each and every Servlet must mandatorily be configured with three different properties namely
  1. Servlet Name (unique)
  2. URL Pattern (Unique)
  3. FQCN

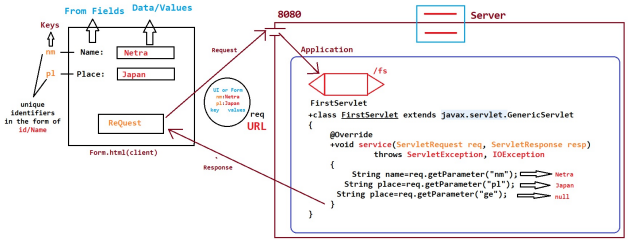


Syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
<servlet-mapping>
<servlet-name>UniqueName</servlet-name>
<url-pattern>/URL</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>UniqueName</servlet-name>
<servlet-class>FQCN</servlet-class>
</servlet>
</web-app>
```

25. UI or Form data ?

▼ Ans



- The data which is entered by the enducer (client) on a Form Page and is submitted to the Server in the form of Key and Value Pair is known as UI or FORM DATA.
- All the keys associated with Form Page are represented as Unique Identifier in the form of id or name.
- The UI or Form Data is always associated with a Request and can be accessed only within the service( ) method. since service( ) is the only method which takes a parameter called ServletRequest and ServletResponse which is responsible for processing the ClientRequest.
- The UI or Form Data can be fetched by using getParameter( ).

```
public String getParameter(String key)

a) In this method, the key is taken as an Argument.
b) If the key is present, then the method returns associated value.
c) If the key is not present, then the method returns NULL,
   but not any error or Exception.
```

- Once a Request is made, the data are carried to the Server as a part of Request Object in the form of Key and Value Pair which is displayed on the URL. Hence, The data present in the request object are UI or From data.

26. Steps to Create a Dynamic Web Application ?

▼ Ans

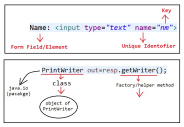
1. Open Eclipse in JEE Perspective.
2. Open Navigator Mode(Click in Window and View And Click on Navigator).
3. Right Click with Navigator Mode and Create a new Dynamic Web Project and Name it.
4. Generate a web.xml [Right Click on Project and Select Java EE Tools and Click on generate Deployment Description Stub]. —> Generated inside WEB-INF folder.
5. Add servlet-api.jar into lib folder which is present in WEB-INF folder.
6. Create a Resource
    a. HTML Resource ⇒ Filename.html ⇒ Select Webcontent folder and Hit ctrl N and type ht to Create a new HTML file.

b. Servlet Resource ⇒ ServletName.java ⇒ Select src folder and hit crtl N to Create a new package structure. Select the Application Name and hit control N to Create a new class.

7. Configure a Resource
   a. By using web.xml.
   b. By using Annotation.
8. Deploy a Resource ⇒ webapps folder of Apache Tomcat Server
   a. Create a new folder inside webapps and name it (consider as web Application).
   b. Web Application ⇒ 1.(WEB-INF (classes, lib, web.xml)) 2.(Filename.html).

27. Code for UI or Form data Using web.xml ?

▼ Ans



▼ Form.html (HTML Code)

```
<!DOCTYPE html>
<html>
<meta charset="ISO-8859-1">
<body bgcolor="cyan">

<form action="fs">
Name: <input type="text" name="nm">
<br><br>
Place: <input type="text" name="pl">
<br><br>
<input type="submit" value="Request">
</form>
</body>
</html>
```

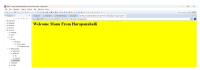
▼ FirstServet.java (Servlet Code)

```
package org.Manukm.UI_FormApp;
import java.io.*;
import javax.servlet.*;
public class FirstServlet extends GenericServlet
{
  @Override
  public void service(ServletRequest req, ServletResponse resp)
      throws ServletException, IOException
  {
    String name=req.getParameter("nm");
    String place=req.getParameter("pl");
    //Display Response on new HTML File
    PrintWriter out=resp.getWriter();
    out.println("<html><body bgcolor='yellow'>"
        +"<h1>Welcome "+name+" From "+place+"</h1>"+"</body></html>");
    //Presentation Logic
    out.close();
  }
}
```

▼ web.xml (XML Code)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http:/
  <display-name>UI_Form</display-name>
```

```
  <welcome-file-list>
    <welcome-file>Form.html</welcome-file>
  </welcome-file-list>

  <servlet-mapping>
  <servlet-name>FirstServ</servlet-name>
  <url-pattern>/fs</url-pattern>
  </servlet-mapping>

  <servlet>
 <servlet-name>FirstServ</servlet-name>
 <servlet-class>org.Manukm.UI_FormApp.FirstServlet</servlet-class>
  </servlet>

</web-app>
```



28. Code for UI or Form data Using Annotation ?

▼ Ans

1. Delete or Don’t generate web.xml
2. Rename Form.html to index.html
3. Add Annotation in FirstServlet.class

▼ index.html (HTML Code)

```
<!DOCTYPE html>
<html>
<meta charset="ISO-8859-1">
<body bgcolor="cyan">
<form action="fs">
Name: <input type="text" name="nm">
<br><br>
Place: <input type="text" name="pl">
<br><br>
<input type="submit" value="Request">
</form>
</body>
</html>
```

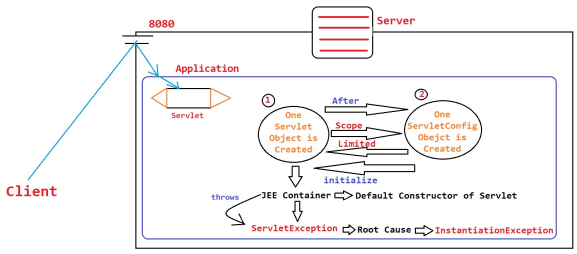
▼ FirstServet.java (Servlet Code)

```
package org.Manukm.UI_FormApp;
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;

@WebServlet("/fs")  //(Using Annotaion)
public class FirstServlet extends GenericServlet
{
  @Override
  public void service(ServletRequest req, ServletResponse resp)
      throws ServletException, IOException
  {
    String name=req.getParameter("nm");
    String place=req.getParameter("pl");
    //Display Response on new HTML File
    PrintWriter out=resp.getWriter();
    out.println("<html><body bgcolor='yellow'>"
        +"<h1>Welcome "+name+" From "+place+"</h1>"+"</body></html>");
    //Presentation Logic
    out.close();
```

```
  }
}
```

29. Servlet Life Cycle ?

▼ Ans



- Servlet gets a life and starts its Lifecycle only when a Servlet Object is created.

▼ Definition

- Servlet Lifecycle depicts or represents the events or phases from Servlet Object Creation until Servlet Object Destruction.

- The entire Servlet Life Cycle is managed by JEE Container.
- There are 4 different Servlet Life Cycle Phases are present namely:

▼ 1). Instantiation or Object Creation Phase

- In this phase, the Servlet Object has to be created.
- Whenever the Client makes a First Request to the Servlet, one Servlet Object is created by the JEE Container by calling the default constructor of Servlet and now Servlet Lifecycle begins.
- If the JEE Container does not find the default constructor of Servlet, then the JEE Container throws an exception called ServletException with a root cause InstantiationException (Servlet Object not Created).
- Immediately after the Servlet Object Creation, one ServletConfig object is created by the JEE Container which is used to initialize the resources of that particular Servlet Object.
- Hence, the scope of ServletConfig object is always limited to that particular Servlet Object.

▼ 2). Initialization Phase

- In this phase, Servlet Object has to be Initialized.
- ServletObject has to be initialized by using init(ServletConfig) method which takes ServletConfig as a parameter which is used to initialize the resources of that particular Servlet Object.
- init() method is always called by the JEE Container only once.
- If this phase fails, then the JEE Container throws an exception called ServletException. (Client Request Processing Failed)

▼ 3). Service Phase

- In this phase, service() method is called which is responsible for processing each and every client request.
- In this phase, the JEE Container creates one request and one response object for each and every client request including the first client request.
- Whenever a client makes a second or subsequent client request to the same Servlet once again, only service() method will be executed but the Servlet object will not going to be created once again.
- By default, Servlet is Multi-threaded. But, It can be made single threaded in two different ways :-

1. By writing a Servlet Class which implements a marker interface by name called SingleThreaded Model. (which is a depricated interface) means out-dated.
2. By making service() method as Synchronized.

- service() method is called by the JEE Container for multiple times.
- If this phase fails, then the JEE Container throws an exception called ServletException.

▼ 4). Destruction Phase

- In this phase, the destroy( ) method is called by the JEE Container to close all the Costly Resources related to application.
- destroy( ) method is called by the JEE Container only once.
- If this phase fails, then the performance of an application decreases.

  Situations of destroy( ):

  - destroy() is called by the JEE Container only once but in two different situations namely:
    - Whenever we close an Application, destroy( ) is called by the JEE Container to close all the Costly Resources related to that Application.
    - Whenever we Redeploy an Application onto the server, destroy( ) is called to close all the previously used Costly Resources related to that Application.

      1. What is Redeployed ? When an application is Redeployed onto the Server ?

         ▼ Ans

         - Whenever we make any changes to an application while the server is in use, then the entire application is redeployed onto the Server.

           ▼ Diff between Redeployment and Deployment

           - If the server is not in use or if the server is stopped, and then we try to make changes to an application then we will not call it as a Redeployment instead it again called Deployment.

- Note:-

1. Servlet Object can either be created or be destroy only by JEE Container by its own implementations. But not by calling destroy() method.
2. It is not good practice to destroy the Servlet Object. Since it is needed or further usage.

30. Code for Servlet Life Cycle Using web.xml ?

▼ Ans

▼ HTML (Form.html)

```
<!DOCTYPE html>
<html>
<meta charset="ISO-8859-1">
<body bgcolor="cyan">
<form action="fs">
Name :<input type="text" name="nm"> <br></br>
Place :<input type="text" name="pl"> <br></br>
<input type="submit" value="Request">
</form>
</body>
</html>
```

▼ Servlet (class FirstServlet)

```
package org.manukm.LifeApp;
import java.io.*;
```

```
import javax.servlet.*;
public class FirstServlet extends GenericServlet
{
  public FirstServlet( )
  {
    System.out.println("Servlet Object is Created");
  }
  @Override
  public void init(ServletConfig config) throws ServletException
  {
    System.out.println("Servlet Object is Initialized");
  }
  @Override
  public void service(ServletRequest req, ServletResponse resp)
      throws ServletException, IOException
  {
    String name=req.getParameter("nm");
    String place=req.getParameter("pl");
    PrintWriter out=resp.getWriter( );
    out.println("<html><body bgcolor='yellow'>"
        +"<h1>User Details: "+name+" from "+place+"</h1>"
        +"</body></html>");// PRESENTATION LOGIC //
    out.close( );
    System.out.println("service( ) is executed");
  }
  @Override
  public void destroy( )
  {
    System.out.println("Closed All Costly Resources");
  }
}
```

▼ XML (web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http:/
<display-name>Life_Cycle_UsingXML</display-name>
<welcome-file-list>
<welcome-file>Form.html</welcome-file>
</welcome-file-list>
<servlet-mapping>
<servlet-name>FirstServ</servlet-name>
<url-pattern>/fs</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>FirstServ</servlet-name>
<servlet-class>org.manukm.LifeApp.FirstServlet</servlet-class>
</servlet>
</web-app>
```

```
INFO: Starting Servlet Engine: Apache Tomcat/8.0.30
Dec 28, 2022 3:15:43 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8050"]
Dec 28, 2022 3:15:43 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-nio-8009"]
Dec 28, 2022 3:15:43 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 310 ms
Servlet Object is Created
Servlet Object is Initialized
service( ) is executed
```

30. Code for Servlet Life Cycle Using Annotation ?

▼ Ans

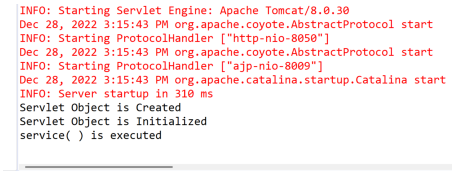
▼ HTML (index.html)

```
<!DOCTYPE html>
<html>
<meta charset="ISO-8859-1">
<body bgcolor="cyan">
<form action="fs">
Name :<input type="text" name="nm"> <br></br>
Place :<input type="text" name="pl"> <br></br>
<input type="submit" value="Request">
```

```
</form>
</body>
</html>
```

▼ Servlet (class FirstServlet)

```
package org.manukm.LifeApp;
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
@WebServlet("/fs")
public class FirstServlet extends GenericServlet
{
  public FirstServlet( )
  {
    System.out.println("Servlet Object is Created");
  }
  @Override
  public void init(ServletConfig config) throws ServletException
  {
    System.out.println("Servlet Object is Initialized");
  }
  @Override
  public void service(ServletRequest req, ServletResponse resp)
      throws ServletException, IOException
  {
    String name=req.getParameter("nm");
    String place=req.getParameter("pl");
    PrintWriter out=resp.getWriter( );
    out.println("<html><body bgcolor='yellow'>"
        +"<h1>User Details: "+name+" from "+place+"</h1>"
        +"</body></html>");// PRESENTATION LOGIC //
    out.close( );
    System.out.println("service( ) is executed");
  }
  @Override
  public void destroy( )
  {
    System.out.println("Closed All Costly Resources");
  }
}
```



30. How many methods present in Servlet Life Cycle ?

▼ Ans

1. init (Servletconfig)
2. service(ServletRequest req , ServletResponse resp)
3. destroy()

31. In how many ways can we create Servlet Object ?

▼ Ans

- Servlet Object can be created in two different ways :-
  1. Whenever a client makes a first request to a Servlet. One Servlet Object created by JEE Container by calling the default constructor of Servlet.
  2. **In case of Load-on-Startup.**

31. LOAD ON STARTUP ?

▼ Ans

- **Servlet gets a life and starts its Lifecycle only when a Servlet Object is created.**
- <load-on-startup> is a tag which is a sub-tag of <Servlet>.
- In case of Load-on-Startup, the JEE Container creates a Servlet Object by calling the default constructor of Servlet **at the time of Server Startup** without waiting for the first client request. So that the delay time made by the first client request can be avoided which helps in **increasing the performance of an Application.**
- In case of Load-on-Startup, only service() will be executed for each and every client request including the first client request.
- Load-on-Startup must mandatorily configured with a positive integer value. But, the JEE Container gives the priority based on **lowest positive integer value**.
- Whenever two Servlets are configured with the same positive integer value, then **sequential execution takes place**.
- Whenever Load-on-Startup is configured with a negative integer value, then the Servlet Object is created based on first client request ignoring Load-on-Startup without throwing any Error or Exception.

32. Code for Servlet Lifecycle in case of load-on-startup using web.xml ?

▼ Ans

▼ HTML (Form.html)

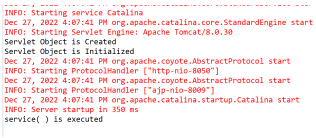
```
<!DOCTYPE html>
<html>
<meta charset="ISO-8859-1">
<body bgcolor="cyan">
<form action="fs">
Name :<input type="text" name="nm"> <br></br>
Place :<input type="text" name="pl"> <br></br>
<input type="submit" value="Request">
</form>
</body>
</html>
```

▼ Servlet (class FirstServlet)

```
package org.manukm.LoadApp;
import java.io.*;
import javax.servlet.*;
public class FirstServlet extends GenericServlet
{
  public FirstServlet( )
  {
    System.out.println("Servlet Object is Created");
  }
  @Override
  public void init(ServletConfig config) throws ServletException
  {
    System.out.println("Servlet Object is Initialized");
  }
  @Override
  public void service(ServletRequest req, ServletResponse resp)
      throws ServletException, IOException
  {
    String name=req.getParameter("nm");
    String place=req.getParameter("pl");
    PrintWriter out=resp.getWriter( );
    out.println("<html><body bgcolor='yellow'>"
        +"<h1>User Details: "+name+"from "+place+"</h1>"
        +"</body></html>");// PRESENTATION LOGIC //
    out.close( );
    System.out.println("service( ) is executed");
  }
  @Override
  public void destroy( )
  {
    System.out.println("Closed All Costly Resources");
  }
}
```

▼ XML (web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http:/
  <display-name>Load_on_startUP_UsingXML</display-name>

   <welcome-file-list>
 <welcome-file>Form.html</welcome-file>
 </welcome-file-list>

 <servlet-mapping>
 <servlet-name>FirstServ</servlet-name>
 <url-pattern>/fs</url-pattern>
 </servlet-mapping>

 <servlet>
 <servlet-name>FirstServ</servlet-name>
 <servlet-class>org.manukm.LoadApp.FirstServlet</servlet-class>
 <load-on-startup>5</load-on-startup>
 </servlet>

</web-app>
```



32. Code for Servlet Lifecycle in case of load-on-startup using Annotation ?

▼ Ans

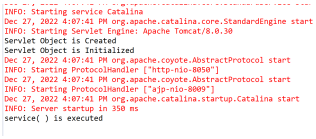
▼ HTML (index.html)

```
<!DOCTYPE html>
<html>
<meta charset="ISO-8859-1">
<body bgcolor="cyan">
<form action="fs">
Name :<input type="text" name="nm">
<br></br>
Place :<input type="text" name="pl">
<br></br>
<input type="submit" value="Request">
</form>
</body>
</html>
```

▼ Servlet (class FirstServlet)

```
package org.manukm.LoadApp;
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;

@WebServlet(urlPatterns="/fs", loadOnStartup=5)
public class FirstServlet extends GenericServlet
{
  public FirstServlet( )
  {
    System.out.println("Servlet Object is Created");
  }
  @Override
  public void init(ServletConfig config) throws ServletException
  {
    System.out.println("Servlet Object is Initialized");
  }
  @Override
```

```
  public void service(ServletRequest req, ServletResponse resp)
      throws ServletException, IOException
  {
    String name=req.getParameter("nm");
    String place=req.getParameter("pl");
    PrintWriter out=resp.getWriter( );
    out.println("<html><body bgcolor='yellow'>"
        +"<h1>User Details: "+name+" from "+place+"</h1>"
        +"</body></html>");// PRESENTATION LOGIC //
    out.close( );
    System.out.println("service( ) is executed");
  }
  @Override
  public void destroy( )
  {
    System.out.println("Closed All Costly Resources");
  }
}
```



32. What is Redeployed ? When an application is Redeployed onto the Server ?

▼ Ans

- Whenever we make any changes to an application while the server is in use, then the entire application is redeployed onto the Server.

▼ Diff between Redeployment and Deployment

- If the server is not in use or if the server is stopped, and then we try to make changes to an application then we will not call it as a Redeployment instead it again called Deployment.

33. Http Post ?

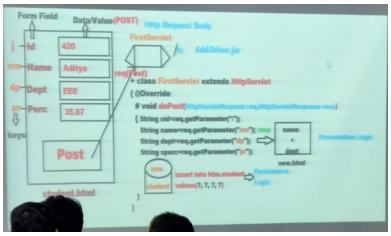
▼ Ans

- **Post request is used to post some contents or data from the Client to the Server.**
- **Post request deals with Unlimited data.**
- **Post request is Non-idempotent.**
  - **Non-idempotent :- means Not specific to any Resource.**
- **Post request cannot be Bookmarked.**
  - **Bookmarked :- means saved**
- **In case of Post request, the data are carried to the Server as a part of Http Request Body which is not displayed even to the Enducer(Client). Hence the data are Secured.**
- **Whenever we deal with N number of data, then the type of request is Post.**

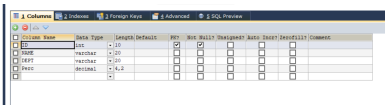
34. Requirement for Post Request ?

▼ Ans

▼ Requirement



▼ Table Creation



▼ HTML Code (Student.html)

```
<!DOCTYPE html>
<html>
<title>Insert title here</title>
<body bgcolor="cyan">
  <form action="fs" method="post">
    <table>
      <tr>
        <td>Id:</td>
        <td><input type="text" name="i"></td>
      </tr>
      <tr>
        <td>Name:</td>
        <td><input type="text" name="nm"></td>
      </tr>
      <tr>
        <td>Dept:</td>
        <td><input type="text" name="dp"></td>
      </tr>
      <tr>
        <td>Perc:</td>
        <td><input type="text" name="pr"></td>
      </tr>
      <tr>
        <td></td>
        <td></td>
      </tr>
      <tr>
        <td></td>
        <td><input type="submit" value="Post"></td>
      </tr>
    </table>
  </form>
</body>
</html>
```

▼ Servlet Code (FirstServlet.class)

```
package org.Manukm.PostApp;
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class  FirstServlet extends HttpServlet
{
  @Override
  protected void doPost(HttpServletRequest req,
      HttpServletResponse resp) throws ServletException, IOException
  {
    String sid=req.getParameter("i");
    int id=Integer.parseInt(sid);

    String name=req.getParameter("nm");
    String dept=req.getParameter("dp");
```

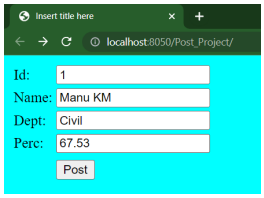
```
    String sperc=req.getParameter("pr");
    double perc=Double.parseDouble(sperc);

    // PRESENTATION LOGIC //
    PrintWriter out=resp.getWriter( );
    out.println("<html><body bgcolor='yellow'>"
        +"<h1>Student Details are: "+name+" from "+dept+
        " Department"+"</h1>"+ "</body></html>");
    out.close( );
    // PERSISTENCE LOGIC //

    Connection con=null;
    PreparedStatement pstmt=null;
    String qry="Insert into btm.student values(?,?,?,?)";
    try
    {
      Class.forName("com.mysql.jdbc.Driver");
      con=DriverManager.getConnection("jdbc:mysql://localhost:3306?user=root&password=admin");
      pstmt=con.prepareStatement(qry);
      // SET THE VALUE FOR PLACEHOLDER BEFORE EXECUTION //
      pstmt.setInt(1,id);
      pstmt.setString(2,name);
      pstmt.setString(3,dept);
      pstmt.setDouble(4,perc);

      pstmt.executeUpdate( );
    }
    catch (ClassNotFoundException | SQLException e)
    {
      e.printStackTrace( );
    }
    finally
    {
      if(pstmt!=null)
      {
        try
        {
          pstmt.close( );
        }
        catch (SQLException e)
        {
          e.printStackTrace( );
        }
      }
      if(con!=null)
      {
        try
        {
          con.close( );
        }
        catch (SQLException e)
        {
          e.printStackTrace( );
        }
      }
    }
  }
}
```

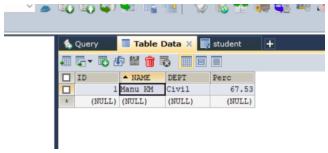
▼ web.xml Code

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http:/
  <display-name>Post_Project</display-name>

  <welcome-file-list>
 <welcome-file>Student.html</welcome-file>
 </welcome-file-list>

 <servlet-mapping>
 <servlet-name>FirstServ</servlet-name>
 <url-pattern>/fs</url-pattern>
 </servlet-mapping>

 <servlet>
 <servlet-name>FirstServ</servlet-name>
 <servlet-class>org.Manukm.PostApp.FirstServlet</servlet-class>
```

```
 </servlet>

</web-app>
```

▼ Output





34. Http Get ?

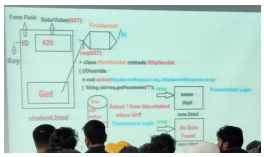
▼ **Ans**

- **Get request is used to get some contents or resources from the Server.**
- **Get request deals with only Limited Data i.e. 1024 Characters.**
- **Get request is Idempotent. that means it is specific to particular resource.**
- **Get request can be Bookmarked. that means it can be saved.**
- **In case of Get request, the data are carried to the Server as a part of Request Object in the form of Key and Value pair which is displayed to the Enducer(Client) in the URL. Hence, the data are not Secured.**
- **Whenever the type of request is not mentioned or configured, then by default the type of request is Get request.**
- **Whenever we deal with Link, then the type of request is Get request.**

34. Requirement for Get Request ? Code to Dynamically save the Data into the Database Server using Post Request ?

▼ Ans

▼ Requirement



▼ HTML Code (Student.html)

```
<!DOCTYPE html>
<html>
<title>Insert title here</title>
<body bgcolor="cyan">
<form action="fs" method="get">
Id:<input type="text" name="i">
<br></br>
<input type="submit" value="Get">
</form>
```

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

▼ Servlet Code (FirstServlet.class)

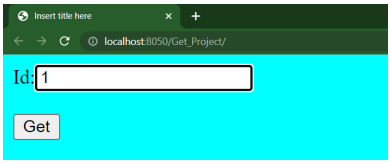
```
package org.Manukm.GetApp;
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class FirstServlet extends HttpServlet
{
  @Override
  protected void doGet(HttpServletRequest req,
      HttpServletResponse resp) throws ServletException, IOException
  {
    String sid=req.getParameter("i");
    int id=Integer.parseInt(sid);
    Connection con=null;
    PreparedStatement pstmt=null;
    ResultSet rs=null;
    String qry="select * from btm.student where id=?";
    try {
      Class.forName("com.mysql.jdbc.Driver");
      con=DriverManager.getConnection("jdbc:mysql://localhost:3306?user=root&password=admin");
      pstmt=con.prepareStatement(qry);
      // SET THE VALUE FOR PLACEHOLDER BEFORE EXECUTION //
      pstmt.setInt(1,id);
      rs=pstmt.executeQuery( );
      // PERSISTENCE LOGIC //
      PrintWriter out=resp.getWriter( );
      if(rs.next( ))
      {
        String name=rs.getString(2);
        String dept=rs.getString(3);
        out.println("<html><body bgcolor='yellow'>"
            +"<h1>Student Details are "+name+" from "+dept+
            " Department"+"</h1>"+"</body></html>");
        // PRESENTATION LOGIC //
        out.close( );
      }
      else
      {
        out.println("<html><body bgcolor='pink'>"
            +"<h1>No Data Found!!!!</h1>"
            +"</body></html>");
        // PRESENTATION LOGIC //
        out.close( );
      }
    } catch (ClassNotFoundException | SQLException e) {
      e.printStackTrace( );
    }
    finally
    {
      if(rs!=null)
      {
        try {
          rs.close( );
        } catch (SQLException e) {
          e.printStackTrace( );
        }
      }
      if(pstmt!=null)
      {
        try {
          pstmt.close( );
        } catch (SQLException e) {
          e.printStackTrace( );
        }
      }
      if(con!=null)
      {
        try {
          con.close( );
        } catch (SQLException e) {
          e.printStackTrace( );
        }
      }
```

```
    }
  }
}
```

▼ web.xml Code

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http:/
  <display-name>Get_Project</display-name>
  <welcome-file-list>
 <welcome-file>Student.html</welcome-file>
 </welcome-file-list>
 <servlet-mapping>
 <servlet-name>FirstServ</servlet-name>
 <url-pattern>/fs</url-pattern>
 </servlet-mapping>
 <servlet>
 <servlet-name>FirstServ</servlet-name>
 <servlet-class>org.Manukm.GetApp.FirstServlet</servlet-class>
 </servlet>
</web-app>
```

▼ Output



35. Cookie ?

▼ Ans

- An object of Cookie will be created when will click on to SAVE button on pop-up button. Cookie is not permanent storage area. It is temporary storage area. The storage area for Cookie object is Browser cheche memory.
- If we click on NEVER an object Request is going to be created. Once we Log Out that request object also destroyed.

---

1. **Registration Form by Diamond ?**

▼ Ans

▼ login.html

```
<!DOCTYPE html>
<html>
<meta charset="ISO-8859-1">
<style>
#d1
{
  border: 2px solid black;
  width: 375px;
  height: 230px;
  margin: auto;
    position: absolute;
    left: 0;right: 0;top: 0;bottom: 0;
}
.in
{
    width:200px;
}
```

```
form
{
    padding: 20px;
}
</style>
<body >
  <div id="d1">
  <center>
  <form action="lo">
  <pre>
 EMAIL-ID   :<input type="text" name="em" class="in">
  <br>
 PASSWORD   :<input type="text" name="ps" class="in">
  </pre>
  <center><input type="submit" value="Login"></center>
  </form>
  <center>or</center>
  <a href="register.html"><button>Register</button></a>
  </center>
  </div>
</body>
</html>
```

▼ loginServlet.java

```
package com.Esmd.reg_App;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.*;
public class loginServlet extends GenericServlet
{
  @Override
  public void service(ServletRequest req, ServletResponse res)
      throws ServletException, IOException
  {
    String email    = req.getParameter("em");
    String password = req.getParameter("ps");
    if((password==""&&email=="")!=true)
    {
      Connection          con = null;
      PreparedStatement pstmt = null;
      ResultSet           rs = null;
      String             qry ="select * from esmd.register where emailid=? and password=?";
      try {
        Class.forName("com.mysql.jdbc.Driver");
        con   = DriverManager.getConnection("jdbc:mysql://localhost:3306?user=root&password=admin");
        pstmt = con.prepareStatement(qry);
        pstmt.setString(1, email);
        pstmt.setString(2, password);
        rs    = pstmt.executeQuery();
        if(rs.next())
        {
          PrintWriter out = res.getWriter();
          out.println("<center><h1>DETAILS ARE:</h1></center>");
          out.println("<center><h3>NAME        : "+rs.getString(2)+"</h3></center>");
          out.println("<center><h3>PHONE       : "+rs.getLong(3)+"</h3></center>");
          out.println("<center><h3>COLLEGE     : "+rs.getString(4)+"</h3></center>");
          out.println("<center><h3>DEPT        : "+rs.getString(5)+"</h3></center>");
          out.println("<center><h3>YEAR        : "+rs.getInt(6)+"</h3></center>");
          out.println("<center><h3>10TH PERC   : "+rs.getDouble(7)+"</h3></center>");
          out.println("<center><h3>12TH PERC   : "+rs.getDouble(8)+"</h3></center>");
          out.println("<center><h3>DEGREE PERC : "+rs.getDouble(9)+"</h3></center>");
          out.close();
        }
        else
        {
          PrintWriter out = res.getWriter();
          out.println("<center><h1>YOUR NOT REGISTERED</h1></center>");
          out.close();
        }
      }
      catch (ClassNotFoundException | SQLException e)
```

```
      {
        e.printStackTrace();
      }
      finally
      {
        if(rs!=null)
        {
          try {
            rs.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        if(pstmt!=null)
        {
          try {
            pstmt.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        if(con!=null)
        {
          try {
            con.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }

        }
      }
    }
    else
    {
      PrintWriter out = res.getWriter();
      out.println("<center><h1>ENTER THE VALUE!!!</h1></center>");
      out.close();
    }
  }
}
```

▼ register.html

```
<!DOCTYPE html>
<html>
<meta charset="ISO-8859-1">
<style>
#box
{
 border:2px solid black;
 width: 350px;
 padding: 20px;
 margin: auto;
}
.in
{
width: 200px
}
</style>
<body>
<center><h1>REGISTERATION FORM</h1></center>
<div id="box">
<form action="reg">
<pre>
   EMAIL ID    :<input type="text" class="in" name="id"> <br><br>
   NAME        :<input type="text" class="in" name="nm"> <br><br>
   PHONE.NO    :<input type="text" class="in" name="pn"> <br><br>
   COLLEGE     :<input type="text" class="in" name="co"> <br><br>
   DEPT        :<input type="text" class="in" name="de"> <br><br>
   YEAR(PASS)  :<input type="text" class="in" name="yr"> <br><br>
   10th %      :<input type="text" class="in" name="teper"> <br><br>
   12th %      :<input type="text" class="in" name="twper"> <br><br>
   DEGREE CGPA :<input type="text" class="in" name="deper"> <br><br>
   PASSWORD    :<input type="password" class="in" name="pas"> <br><br>
   <label>GENDER     :</label><input type="radio" name="ge" value="MALE">Male <input type="radio" name="ge" value="FEMALE">Fem
</pre>
<center><input type="submit" value="Register"></center>
<center>or</center>
```

```
</form>
<center><a href="login.html"><button>Login</button></a></center>
</div>
</body>
</html>
```

▼ reg_Servlet.java

```
package com.Esmd.reg_App;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.Map;

import javax.servlet.*;
public class reg_Servlet extends GenericServlet
{
  @Override
  public void service(ServletRequest req, ServletResponse res)
      throws ServletException, IOException
  {
    //Store the value in the ArrayList to Verify it,not present null value;
    Enumeration<String> el=req.getParameterNames();
    ArrayList  <String> al=new ArrayList<String>();
    while(el.hasMoreElements())
    {
      al.add(req.getParameter(el.nextElement()));
    }
    boolean tr=check(al);
    if(tr)
    {
      //check for already present Using ResultSet

      //getting UI/FORM data from register.html
      String emailid =req.getParameter("id");
      String name    =req.getParameter("nm");
      Long   phone   =Long.parseLong(req.getParameter("pn"));
      String college =req.getParameter("co");
      String dept    =req.getParameter("de");
      int    year    =Integer.parseInt(req.getParameter("yr"));
      double tenperc =Double.parseDouble(req.getParameter("teper"));
      double twperc  =Double.parseDouble(req.getParameter("twper"));
      double degperc =Double.parseDouble(req.getParameter("deper"));
      String pass    =req.getParameter("pas");
      String gender  =req.getParameter("ge");
      //storing to DATABASE esmd.register
      Connection          con = null;
      PreparedStatement pstmt = null;
      ResultSet           rs = null;
      String            qry1 = "select emailid,phone from esmd.register where emailid=? or phone=?";
      String            qry2 = "insert into esmd.register values(?,?,?,?,?,?,?,?,?,?,?)";
      try  {
        Class.forName("com.mysql.jdbc.Driver");
        con          = DriverManager.getConnection("jdbc:mysql://localhost:3306?user=root&password=admin");
        pstmt        = con.prepareStatement(qry1);
        pstmt.setString(1,emailid);
        pstmt.setLong  (2, phone);
        rs           = pstmt.executeQuery();
        if(rs.next()==false)
        {
           pstmt        = con.prepareStatement(qry2);
           pstmt.setString(1,emailid);
           pstmt.setString(2,name )  ;
           pstmt.setLong  (3,phone)  ;
           pstmt.setString(4,college);
           pstmt.setString(5,dept)   ;
           pstmt.setInt   (6,year)   ;
           pstmt.setDouble(7,tenperc);
           pstmt.setDouble(8,twperc) ;
           pstmt.setDouble(9,degperc);
           pstmt.setString(10,gender);
           pstmt.setString(11,pass);
```

```
            pstmt.execute();
            PrintWriter out= res.getWriter();
            out.println("<center><h1>YOUR SUCCESSFULLY REGISTERED!!!.</h1></center>");
            out.close();
          }
          else
          {
            PrintWriter out= res.getWriter();
            out.println("<center><h1>YOUR ALREADY REGISTERED.JUST LOGIN.</h1></center>");
            out.close();
          }
        }
        catch(ClassNotFoundException | SQLException e)
        {
        System.out.println(e);
        }
        finally
        {
          if(pstmt!=null)
          {
            try {
              pstmt.close();
            }
            catch(SQLException e)
            {
              e.printStackTrace();
            }
          }
          if(con!=null)
          {
            try {
              con.close();
            }
            catch(SQLException e)
            {
              e.printStackTrace();
            }
          }
          if(rs!=null)
          {
            try {
              rs.close();
            }
            catch(SQLException e)
            {
              e.printStackTrace();
            }
          }
        }
      }
      else
      {
        PrintWriter out= res.getWriter();
        out.println("<center><h1>Please enter All Value</h1></center>");
        out.close();
      }
    }

    private boolean check(ArrayList<String> al)
    {
      for(String i:al)
      {
        if(i.equals(""))
        {
          return false;
        }
      }
      return true;
    }
}
```

▼ web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="h
  <display-name>register_proj</display-name>
  <welcome-file-list>
  <welcome-file>login.html</welcome-file>
```

```
  <welcome-file>register.html</welcome-file>
  </welcome-file-list>
  <servlet-mapping>
  <servlet-name>reg_servlet</servlet-name>
  <url-pattern>/reg</url-pattern>
  </servlet-mapping>
  <servlet>
  <servlet-name>reg_servlet</servlet-name>
  <servlet-class>com.Esmd.reg_App.reg_Servlet</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>loginServ</servlet-name>
  <url-pattern>/lo</url-pattern>
  </servlet-mapping>
  <servlet>
   <servlet-name>loginServ</servlet-name>
   <servlet-class>com.Esmd.reg_App.loginServlet</servlet-class>
  </servlet>
</web-app>
```

2. **Registration Form by Diamond ? (Loose Coupling Code with MySQL and Oracle using web.xml) ?**

▼ **Ans**

▼ **login.jsp**

```
<html>
<head>
<meta charset="ISO-8859-1">
<link rel="stylesheet" href="Instagramcss.css">
</head>
<body>
<div class="login-box">
    <h2>Login</h2>
    <form action="login" method="post">
        <div class="user-box">
            <input type="email" name="em" class="in" required="">
            <label>Email</label>
        </div>
        <div class="user-box">
            <input type="password" name="ps" class="in" required="">
            <label>Password</label>
        </div>
        <input type="submit" value="Login" id="A">
            <span></span>
            <span></span>
            <span></span>
            <span></span>
   </form>
</div>
</html>
```

▼ **loginServlet.java**

```
package org.SJ.CRUDApp;

import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class loginServlet extends HttpServlet {
  @Override
  public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
    String email = req.getParameter("em");
    String password = req.getParameter("ps");
    RequestDispatcher dispatcher = null;

    if ((password == "" && email == "") == false) {
      Connection con = null;
      PreparedStatement pstmt = null;
      ResultSet rs = null;
      String qry = "select * from btm.crud where email=? and pass=?";
      try {
        Class.forName("com.mysql.jdbc.Driver");
```

```
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306?user=root&password=admin");
        pstmt = con.prepareStatement(qry);
        pstmt.setString(1, email);
        pstmt.setString(2, password);
        rs = pstmt.executeQuery();

        loginServlet manu = new loginServlet();
        HttpSession session = req.getSession();
        session.setAttribute("rs", rs);
        session.setAttribute("con", con);
        session.setAttribute("pstmt", pstmt);
        session.setAttribute("manu", manu);

        if (rs.next()) {
          dispatcher = req.getRequestDispatcher("Reading.jsp");
          dispatcher.forward(req, res);
        } else {

          PrintWriter out = res.getWriter();
          out.println(
              "<h1 style='color: white; margin-left: 250px; margin-top: 100px;'>Sorry Your not Registered yet!! Please Kindly
                  + "<a href='register.jsp'>REGISTER</a></h1>");
          dispatcher = req.getRequestDispatcher("login.jsp");
          dispatcher.include(req, res);
          out.close();
        }
      } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
      } finally {
        if (rs != null) {
          try {
            pstmt.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        if (con != null) {
          try {
            con.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        if (pstmt != null) {
          try {
            rs.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
      }
    } else {
      PrintWriter out = res.getWriter();
      out.println(
          "<h1 style='color: white; margin-left: 250px; margin-top: 100px;'>Please Kindly Fill All the Form Feilds !!! </h1>")
      dispatcher = req.getRequestDispatcher("login.jsp");
      dispatcher.forward(req, res);
      out.close();
    }
  }
}
```

▼ **resgister.jsp**

```
<%@page import="java.sql.ResultSet"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta charset="ISO-8859-1">
<link rel="stylesheet" href="RegistrationCss.css">
</head>
<body>
  <div class="login-box">
    <h2>Registeration</h2>
    <form action="register" method="post">
      <div id="Manu">
        <div id="card1">
```

```
          <div class="user-box">
            <input type="email" class="in" name="id" required=""> <label>Email</label>
          </div>

          <div class="user-box">
            <input type="text" class="in" name="pas" required=""> <label>Password</label>
          </div>

          <div class="user-box">
            <input type="text" class="in" name="nm" required=""> <label>Name</label>
          </div>

          <div class="user-box">
            <input type="number" class="in" name="pn" required=""> <label>Phone</label>
          </div>

          <div class="user-box">
            <input type="text" class="in" name="co" required=""> <label>College</label>
          </div>

          <div class="manu">
            <label for="">Gender :</label> <input type="radio" name="ge"
              value="MALE">Male <input type="radio" name="ge"
              value="MALE">Female <input type="radio" name="ge"
              value="MALE">Other
          </div>

        </div>

        <div id="card2">

          <div class="user-box">
            <input type="text" class="in" name="de" required=""> <label>Department</label>
          </div>

          <div class="user-box">
            <input type="text" class="in" name="yr" required=""> <label>Pass
              Out Year</label>
          </div>

          <div class="user-box">
            <input type="text" class="in" name="teper" required=""> <label>10th
              Percentege</label>
          </div>

          <div class="user-box">
            <input type="text" class="in" name="twper" required=""> <label>12th
              Percentege</label>
          </div>

          <div class="user-box">
            <input type="text" class="in" name="deper" required=""> <label>Degree
              CGPA</label>
          </div>
          <div class="manu">
            <label for="">Select DataBase :</label> <input type="radio"
              name="db" value="a">SQL <input type="radio" name="db"
              value="b">Oracle
          </div>
        </div>
      </div>
      <div id="Manu1">
        <input type="submit" value="REGISTER" id="A"> <span></span>
        <span></span> <span></span> <span></span>
      </div>
  </div>
  </form>
  </div>
</body>
</html>
```

▼ **Reg_Servlet.java**

```
package org.SJ.CRUDApp;

import java.io.IOException;
import java.io.*;
import java.sql.*;
```

```
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Reg_Servlet extends HttpServlet {
  @Override
  public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
    // Store the value in the ArrayList to Verify it,not present null value;
    Enumeration<String> el = req.getParameterNames();
    ArrayList<String> al = new ArrayList<String>();
    RequestDispatcher dispatcher = null;

    while (el.hasMoreElements()) {
      al.add(req.getParameter(el.nextElement()));
    }
    boolean tr = check(al);
    if (tr) {
      // getting UI/FORM data from register.html
      String emailid = req.getParameter("id");
      String name = req.getParameter("nm");
      Long phone = Long.parseLong(req.getParameter("pn"));
      String college = req.getParameter("co");
      String dept = req.getParameter("de");
      int year = Integer.parseInt(req.getParameter("yr"));
      double tenperc = Double.parseDouble(req.getParameter("teper"));
      double twperc = Double.parseDouble(req.getParameter("twper"));
      double degperc = Double.parseDouble(req.getParameter("deper"));
      String pass = req.getParameter("pas");
      String gender = req.getParameter("ge");
      String Database = req.getParameter("db");

      if (Database != null) {
        ServletConfig conf = getServletConfig();
        String url = "", dr = "", qry1 = "", qry2 = "", user = "", passw = "";

        if (Database.equals("a")) {// My SQL
          url = conf.getInitParameter("url1");
          dr = conf.getInitParameter("dr1");
          qry1 = "select email,pass from btm.crud where email=? or pass=?";
          qry2 = "insert into btm.crud values(?,?,?,?,?,?,?,?,?,?,?)";
        } else if (Database.equals("b")) {// Oracle
          url = conf.getInitParameter("url2");
          dr = conf.getInitParameter("dr2");
          user = conf.getInitParameter("user");
          passw = conf.getInitParameter("pass1");
          qry1 = "select email,pass from crud where email=? or pass=?";
          qry2 = "insert into crud values(?,?,?,?,?,?,?,?,?,?,?)";
        }
        // storing to DATABASE
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        PrintWriter out = res.getWriter();
        try {
          Class.forName(dr);
          if (Database.equals("a"))
            con = DriverManager.getConnection(url);
          else
            con = DriverManager.getConnection(url, user, passw);
          pstmt = con.prepareStatement(qry1);
          pstmt.setString(1, emailid);
          pstmt.setString(2, pass);
          rs = pstmt.executeQuery();
          // check if for already present Using ResultSet
          if (rs.next() == false) {
            pstmt = con.prepareStatement(qry2);
            pstmt.setString(1, emailid);
            pstmt.setString(2, name);
            pstmt.setLong(3, phone);
            pstmt.setString(4, college);
            pstmt.setString(5, dept);
            pstmt.setInt(6, year);
            pstmt.setDouble(7, tenperc);
            pstmt.setDouble(8, twperc);
            pstmt.setDouble(9, degperc);
            pstmt.setString(10, gender);
            pstmt.setString(11, pass);
            pstmt.execute();

            out.println(
                "<h1 style='color: white; margin-left: 350px; margin-top: 20px;'>Congradulation!! You Successfully Registered<
```

```
                    + "<h1 style='color: white; margin-left: 380px; margin-top: 20px;'>Please Kindly Login for Further Operati
            dispatcher = req.getRequestDispatcher("login.jsp");
            dispatcher.include(req, res);
            out.close();
          } else {
            out.println(
                "<h1 style='color: white; margin-left: 450px; margin-top: 100px;'>You Already Registered!! Just Login</h1>");
            dispatcher = req.getRequestDispatcher("login.jsp");
            dispatcher.include(req, res);
            out.close();
          }
        } catch (ClassNotFoundException | SQLException e) {
          System.out.println(e);
        } finally {
          if (rs != null) {
            try {
              pstmt.close();
            } catch (SQLException e) {
              e.printStackTrace();
            }
          }
          if (con != null) {
            try {
              con.close();
            } catch (SQLException e) {
              e.printStackTrace();
            }
          }
          if (pstmt != null) {
            try {
              rs.close();
            } catch (SQLException e) {
              e.printStackTrace();
            }
          }
        }
      } else {
        dispatcher = req.getRequestDispatcher("SelectDatabase.jsp");
        dispatcher.forward(req, res);
      }
    } else {
      dispatcher = req.getRequestDispatcher("FillAllFieldsRegister.jsp");
      dispatcher.forward(req, res);
    }
  }
  private boolean check(ArrayList<String> al) {
    for (String i : al) {
      if (i.equals("")) {
        return false;
      }
    }
    return true;
  }
}
```

▼ **FillAllFieldsLogin.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta charset="ISO-8859-1">
<link rel='stylesheet' href='NotRegistered.css'>
</head>
<body>
<div>
 <h1>Please Kindly Fill All the Form Feilds !!!</h1>
 <button><a href='login.jsp'>Back</a></button>
</div>
</body>
</html>
```

▼ **FillAllFieldsRegister.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta charset="ISO-8859-1">
<link rel='stylesheet' href='NotRegistered.css'>
</head>
<body>
<div>
 <h1>Please Kindly Fill All the Form Feilds !!!</h1>
 <button><a href='register.jsp'>Back</a></button>
</div>
</body>
</html>
```

▼ **Reading.jsp**

```
<%@page import="java.sql.*"%>
<%@page import="org.SJ.CRUDApp.loginServlet"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta charset="ISO-8859-1">
<link rel='stylesheet' href='ReadingCss.css'>
</head>
<body>
  <%
    ResultSet rs = (ResultSet) session.getAttribute("rs");
    Connection con = (Connection) session.getAttribute("con");
    PreparedStatement pstmt = (PreparedStatement) session.getAttribute("pstmt");
    loginServlet manu = (loginServlet) session.getAttribute("manu");

    if(manu!=null)
    {%>
  <div class='login-box'>
    <h1>Hi, <%=rs.getString("Name")%></h1>
    <br>
    <table>
      <tr>
        <td><label class='A'>Email</label></td>
        <td><label class='B'>:<%=rs.getString("Email")%></label></td>
      </tr>
      <tr>
        <td><label class='A'>Phone</label></td>
        <td><label class='B'>:<%=rs.getString("PNumber")%></label></td>
      </tr>
      <tr>
        <td><label class='A'>Department</label></td>
        <td><label class='B'>:<%=rs.getString("Dept")%></label></td>
      </tr>
      <tr>
        <td><label class='A'>College</label></td>
        <td><label class='B'>:<%=rs.getString("Collage")%></label></td>
      </tr>
      <br>
      <br>
      <tr>
        <td><label class='A'>Year</label></td>
        <td><label class='B'>:<%=rs.getString("year")%></label></td>
      </tr>
      <tr>
        <td><label class='A'>10th Per</label></td>
        <td><label class='B'>:<%=rs.getString("Per10")%></label></td>
      </tr>
      <tr>
        <td><label class='A'>12th Per</label></td>
        <td><label class='B'>:<%=rs.getString("Perc12")%></label></td>
      </tr>
      <tr>
        <td><label class='A'>DegreeCGPA</label></td>
        <td><label class='B'>:<%=rs.getString("DegreePer")%></label></td>
      </tr>
      <tr>
        <td><label class='A'><a href="">Update</a></label></td>
        <td><label class='A'><a href="">Delete</a></label></td>
```

```
      </tr>
      <%if (rs != null) {
        try {
          pstmt.close();
        } catch (SQLException e) {
          e.printStackTrace();
        }
      }
      if (con != null) {
        try {
          con.close();
        } catch (SQLException e) {
          e.printStackTrace();
        }
      }
      if (pstmt != null) {
        try {
          rs.close();
        } catch (SQLException e) {
          e.printStackTrace();
        }
      }
      %>
      <%}
    else
    {
        if (rs != null) {
          try {
            pstmt.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        if (con != null) {
          try {
            con.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        if (pstmt != null) {
          try {
            rs.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        response.sendRedirect("login.jsp");
      }%>

    </table>

  </div>
  </div>
</body>
</html>
```

▼ **SelectDatabase.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta charset="ISO-8859-1">
<link rel='stylesheet' href='NotRegistered.css'>
</head>
<body>
<div>
<h1>You did not Select Data-Base Server!! Please Kindly Select Data-Base Server</h1>
</div>
</body>
</html>
```

▼ **Instagramcss.css**

```
@charset "ISO-8859-1";
body{
    margin:0;
    padding:0;
    font-family: sans-serif;
    background: linear-gradient(#141e30, #243b55);
}
.login-box{
    position: absolute;
    top: 50%;
    left: 50%;
    width: 400px;
    padding: 40px;
    transform: translate(-50%, -50%);
    background: rgba(0,0,0,.5);
    box-sizing: border-box;
    box-shadow: 0 15px 25px rgba(0,0,0,.6);
    border-radius: 10px;
}
.login-box h2{
    margin: 0 0 30px;
    padding: 0;
    color: #fff;
    text-align: center;
}
.login-box .user-box{
    position: relative;
}
.login-box .user-box input{
    width: 100%;
    padding: 10px 0;
    font-size: 16px;
    color: #fff;
    margin-bottom: 30px;
    border: none;
    border-bottom: 1px solid #fff;
    outline: none;
    background: transparent;
}
.login-box .user-box label{
    position: absolute;
    top: 0;
    left: 0;
    padding: 10px 0;
    font-size: 16px;
    color: #fff;
    pointer-events: none;
    transition: .5s;
}
.login-box .user-box input:focus ~ label,
.login-box .user-box input:valid ~ label {
                        top: -20px;
                        left: 0;
                        color: #03e9f4;
                        font-size: 12px;
}
.login-box form #A{
                    position: relative;
                    display: inline-block;
                    padding: 10px 20px;
                    color: black;
                     font-size: 16px;
                     text-decoration: none;
                     text-transform: uppercase;
                     overflow: hidden;
                      transition: .5s;
                      margin-top: 40px;
                    letter-spacing: 4px;
                     margin-left: 90px;
}
.login-box #A:hover{
    background: #03e9f4;
    color: #fff;
    border-radius: 5px;
    box-shadow: 0 0 5px #03e9f4,
                0 0 25px #03e9f4,
                0 0 50px #03e9f4,
                0 0 100 #03e9f4;
}
```

▼ **NotRegistered.css**

```
@charset "ISO-8859-1";
*{
    padding: 0%;
    margin: 0%;
    box-sizing: border-box;
}
body{
    height: 100vh;
    width: 100%;
    background: linear-gradient(#141e30, #243b55);
    display: flex;
    justify-content: center;
    align-items: center;
}
h1{
    font-family: Arial, Helvetica, sans-serif;
    color: white;
}
button{
             position: relative;
                    display: inline-block;
                    padding: 10px 20px;
                    color: black;
                     font-size: 20px;
                     text-decoration: none;
                     text-transform: uppercase;
                     overflow: hidden;
                      transition: .5s;
                      margin-top: 40px;
                    letter-spacing: 4px;
                     margin-left: 330px;
}
button:hover{
   background: #03e9f4;
    color: #fff;
    border-radius: 5px;
    box-shadow: 0 0 5px #03e9f4,
                0 0 25px #03e9f4,
                0 0 50px #03e9f4,
                0 0 100 #03e9f4;
}
button>a{
    font-family: Arial, Helvetica, sans-serif;
    text-decoration: none;
    font-size: 15px;
    font-weight: bold;
    color: black;
}
```

▼ **ReadingCss.css**

```
html{
    height: 100%;
}
body{
    height: 100vh;
    width: 100%;
    margin:0;
    padding:0;
    font-family: sans-serif;
    background: linear-gradient(#141e30, #243b55);
}
.login-box h1{
    margin-top: 40px;
   margin-right: 40px;
    padding: 0;
    color: #fff;
    text-align: center;
}
table{
    position: absolute;
    top: 52%;
    left: 50%;
```

```
    height: 80vh;
    width: 600px;
    padding: 10px;
    transform: translate(-50%, -50%);
    background: rgba(0,0,0,.5);
    box-sizing: border-box;
    box-shadow: 0 25px 25px rgba(0,0,0,.6);
    border-radius: 10px;
}
table .A{
    color: white;
    margin-left: 120px;
    font-size: 20px;
    font-family: Arial, Helvetica, sans-serif;

}
table .B{
    color: white;
    margin-right: 20px;
    font-size: 22px;
    font-family: Arial, Helvetica, sans-serif;
}
table .b1{
    margin-left: 120px;

}

table .b2{
    margin-left: 20px;

}
button{
    height: 40px;
    width: 90px;

}
a:hover{
    transform: scale(1.1);
    transition: 0.5s;
    background-color: #03e9f4;
    box-shadow: 0px 0px 5px white;
}
button>a{
    font-family: Arial, Helvetica, sans-serif;
    text-decoration: none;
    font-size: 15px;
    font-weight: bold;
    color: black;

}
a{
text-decoration: none;
    color: white;
     margin-right: 120px;
     font-size: 30px;

}
```

▼ **RegistrationCss.css**

```
@charset "ISO-8859-1";
html{
    height: 100%;
}
body{
    height: 200px;
    width: 100%;
    margin:0;
    padding:0;
    font-family: sans-serif;
    background: linear-gradient(#141e30, #243b55);
}
.login-box{
            position: absolute;
            top: 50%;
            left: 50%;
            height: 90vh;
            width: 1000px;
```

```
            padding: 40px;
            transform: translate(-50%, -50%);
            background: rgba(0,0,0,.5);
            box-sizing: border-box;
            box-shadow: 0 15px 25px rgba(0,0,0,.6);
            border-radius: 10px;
}
#Manu{
    height: 400px;
    width: 900px;
    display: flex;
    justify-content: space-around;
    align-items: center;
}
#card1{
    height: 200px;
    width: 300px;
    margin-bottom: 200px;
}
#card2{
    height: 200px;
    width: 300px;
    margin-bottom: 200px;

}
.manu{
    color: white;
}
#Manu1{
    margin-left: 350px;
}
.login-box h2{
            margin: 0 0 30px;
            padding: 0;
            color: #fff;
            text-align: center;
}
.login-box .user-box{
                    position: relative;
}
.login-box .user-box input{
                        width: 100%;
                        padding: 10px 0;
                        font-size: 16px;
                        color: #fff;
                        margin-bottom: 30px;
                        border: none;
                        border-bottom: 1px solid #fff;
                        outline: none;
                        background: transparent;
}
.login-box .user-box label{
    position: absolute;
    top: 0;
    left: 0;
    padding: 10px 0;
    font-size: 16px;
    color: #fff;
    pointer-events: none;
    transition: .5s;
}
.login-box .user-box input:focus ~ label,
.login-box .user-box input:valid ~ label {
                       top: -20px;
                       left: 0;
                       color: #03e9f4;
                       font-size: 12px;
}
.login-box form #A{
                    position: relative;
                    display: inline-block;
                    padding: 10px 20px;
                    color: black;
                     font-size: 16px;
                     text-decoration: none;
                     text-transform: uppercase;
                     overflow: hidden;
                      transition: .5s;
                      margin-top: 40px;
                    letter-spacing: 4px;
}
```

```
.login-box #A:hover{
    background: #03e9f4;
    color: #fff;
    border-radius: 5px;
    box-shadow: 0 0 5px #03e9f4,
                0 0 25px #03e9f4,
                0 0 50px #03e9f4,
                0 0 100 #03e9f4;
}
```

▼ **MySQL Table Creation Query**

```
```

▼ **Oracle Table Creation Query**

```
```

3. Registration Form by Diamond ? (Loose Coupling Code using .properties file) ?

▼ Ans