PT Tutorial - 3

Date
Page

Name: Bishal Poudel
Roll no: 191318

1. Differentiate application with applet. Write a simple applet program that demonstrates the event handling.

⇒

Application	Applet
→ Main() method present.	→ Not present.
→ Execution requires JRE	→ Required a browser or applet viewer.
→ called as stand-alone application as application can be executed from command prompt.	→ Requires some third party tool like a browser to execute.
→ can access any data or software available on the system.	→ cannot access any thing on the system except browser's services.
→ doesn't require any security.	→ Requires highest security for the sys. as they are untrusted.

```
import java.applet.*;          /* Applet program to perform
import java.awt.*;                addition of two user input nums.*/
import java.awt.event.*;
public class UserInputDemo extends Applet implements
ActionListner {
    int a=0, b=0;
    String str1, str2, str;
```

```
TextField T1;
TextField T2;
TextField T3;
Button btnAdd;
public void init() {
    T1 = new TextField(10);
    T2 = new TextField(10);
    T3 = new TextField(10);
    btnAdd = new Button("Add");
    T1.setText("0");
    T2.setText("0");
    T3.setText("0");
    add(T1);
    add(T2);
    add(T3);
    add(btnAdd);
    btnAdd.addActionListner(this);
}
@Override
public void actionPerformed(ActionEvent ae) {
    str1 = T1.getText();
    a = Integer.parseInt(str1);
    str2 = T2.getText();
    b = Integer.parseInt(str2);
    T3.setText("sum = " + (a+b));
}
}
```

Date
Page

2. What are the five different methods of applet life cycle? Mention the steps involving conversation of applet to application with suitable example program.

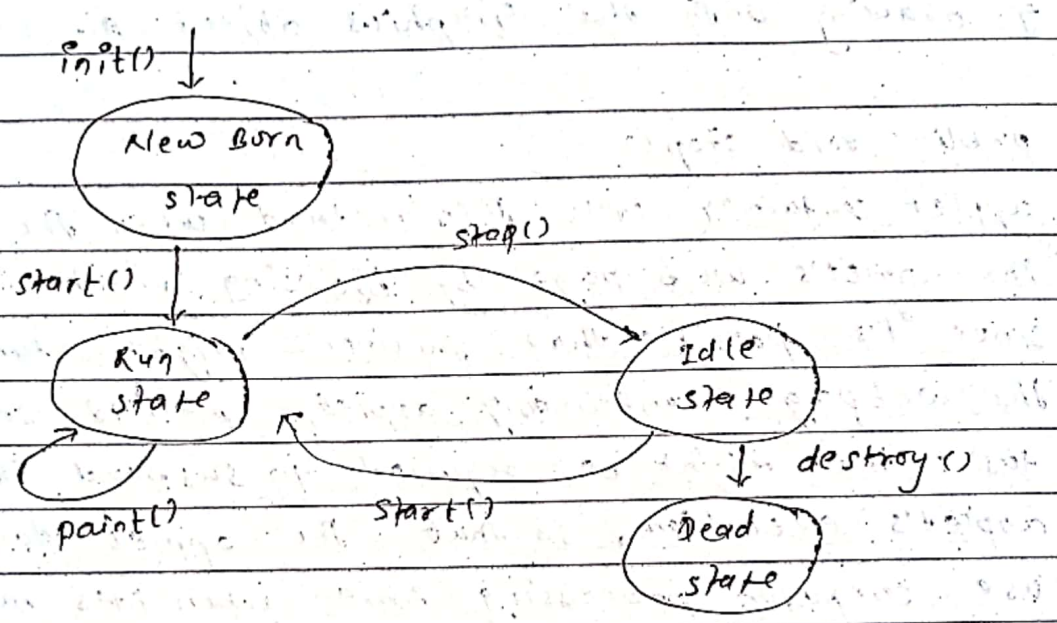


fig. Applet life cycle.

(a) ▷ public void init()

→ called once by applet-container when the applet is loaded for execution. This method initializes an applet. Here perform initializing fields, creating GUI components etc.

(b) public void start()

→ After init() method completes. In addition, if the user browses to another website and later returns to applet's HTML page, method start() is again called. This perform any task that must be completed when the applet is loaded for the first time and that must be performed every time the applet's HTML page is revisited.
eg. starting an animation, starting other threads of execution.

(c) public void paint (Graphics g)

→ called after init() & start() method. It is also called when the applet needs to be repained.
eg. drawing with the graphics object g.

(d) public void stop()

→ applet container calls this method when the user leave the applet's web page by browsing another page. since it's possible that the user might return to the webpage containing applet, method stop() perform tasks that might be required to suspend the applet's execution, so that the applet does not use computer processing time when its not displayed on the screen.
eg. stop the execution of animations and threads.

(e) public void distroy()

→ This method is called when the applet is being removed from memory. This occurs when the user exits the browsing session by closing all the browser windows and may also occur at the browsers discreation when the user has browsed to another web page.
This method perform any tasks that are required to clean up resources allocated to the applet.

Date
Page

Conversion of applet to application:-

(a) Build a java JApplet or Applet assume class name is
(b) change AppletTo Application

(b) Change the name of applet's init() method to same as class name ie. AppletTo Application(), that makes constructor.

(B) Delete void in the header of constructor.

(d) Alter the class header so that it extends Frame rather than Applet.

(e) Create new method called main(). This method should create a Frame object as an instance of the class.

```
public static void main (String[] args) {
    AppletToApplication f = new AppletToApplication();
    f.setSize (300, 300);
    f.setVisible (true);
    f.setLayout (new FlowLayout());
}
```

(d) delete the import for the class Applet.

(g) Add window methods. This also involves adding implements WindowListner and this.addWindowListner(this); to the new constructor method create in (b). and register the handler.

(h) Make sure program does not use any of the methods that are special to Applet class eg. getAudioClip, getCodeBase, getDocumentBase, getImage.

Date
Page

8. Why applet tags are needed to run an applet? Explain the different applet html tags.

=) <applet> tags are HTML tag. It is used to embed java code in HTML. To run any applet we need applet tag so that runtime run and detect java code embded within <applet> tag in HTML.

There are different attributes in <applet> tag.

```
<applet
  [CODEBASE = codebaseURL]
   CODE = appletFile
  [ALT = alternateText]
  [NAME = appletInstanceName]
   WIDTH = pixels
   HEIGHT = pixels
  [ALIGN = alignment]
   [VSPACE = pixles] [HSPACE = pixles]
>
 [<PARAM NAME = AttributeName VALUE = AttributeValue >]
 [HTML displayed in the absence of Java]
</applet>
```

→ CODEBASE is an optional attribute that specifies the base URL of the applet codes which is the directory that will be searched for the applet's executable class file.

→ CODE is required attribute that gives the name of the file containing applet's compiled .class file.

→ ALT tag is an optional attribute used to specify a short text message that should be displayed if the browser recognizes the APPLET tag but can't currently run Java applets.

→ NAME is an optional attribute used to specify a name for the attribute instance. To obtain an applet by name, use getApplet().

→ WIDTH & HEIGHT are required attributs that gives the size in pixels of the applet display area.

→ ALIGN is an optional attributes that specifies allignment of the applet. LEFT, RIGHT, TOP, BOTTOM, MIDDLE etc.

→ VSPACE & HSPACE are optional attributes. VSPACE specifies the space above & below the applet while HSPACE specifies the space on each side of the applet.

→ PARAM NAME & VALUE, It allows us to specify applet-specific argument in an HTML page. Applets access their attributes with the getParameter() method.

Q. How do you pass parameter to the applet? Explain with an example.

⇒ The applet tag in HTML allows us to pass parameter to our applet. To retrieve a parameter, we need to use the getParameter() method. It return the value of specified parameter in the form of string object.

```
/* <applet code="paramDemo" width="400" height="400">
   <param name = "fontName" value = "Times new Roman">
   <param name = "leading" value = "3">
   <param name = "accountEnabled" value = "true">
<~~applet>~~ */ <param name = "fontSize" value = "15">
   </applet> */
import java.awt.*;
import java.applet.*;
public class ParamDemo extends Applet {
    String fontName;
    int fontSize;
    float leading;
    boolean active;

    public void start() {
        String param;
        fontName = getParameter("fontName");
        if (fontName == null)
            fontName = "Not Found";
```

```
param = getParameter ("fontSize");
try {
    if (param != null)
        fontSize = Integer.parseInt (param);
    else
        fontSize = 0;
} catch (NumberFormatException e) {
    fontSize = -1;
}
param = getParameter ("leading");
try {
    if (param != null)
        leading = Float.valueOf (param).floatValue();
    else
        leading = 0;
} catch (NumberFormatException e) {
    leading = -1;
}
param = getParameter ("accountEnabled");
if (param != null)
    active = Boolean.valueOf (param).BooleanValue();
}
public void paint (Graphics g) {
    g.drawString ("Font name: " + fontName, 0, 10);
    g.drawString ("Font size: " + fontSize, 0, 26);
    g.drawString ("Leading: " + leading, 0, 42);
    g.drawString ("Account Active: " + active, 0, 58);
}
}
```

5. What is event handling? Can we handle event without delegation event model? Explain your understanding.

⇒ Event handling is the mechanism that control the event and decides what should happen if an event occur. This mechanism has a code which is known as event handler that is executed when an event occurs.

In most or alsmost we use delegation event model to handle any event. But in some cases we use callback method to handle event. But the Java uses deligation event model. It consists sources and listiners.

→ sources:

Events are genrated from the source. There are various sources like buttons, checkboxes, list, choice, windows etc to generates events.

→ Listners:

Listeners are used for handling the events genrated from the sources. Each of these listners represents interfaces that are responsible for handling events.

To perform event handling, we need to register the source with the listener. The listner is alway listning so that when event occure it handle the event.

6. What is event delegation model? How does it work for event handling in Java?

⇒ Delegation model is event handling model that defines the standard mechanism to generate and handle the events. Its concept is, a source generates an event and sends it to one or more listner. In this scheme, the listner simply waits untill it receives an event. Once event is received, the listner processes the event and then returns.

→ source: source is an object on which event occurs. source is responsible for providing information of the occurred event to its handler. Java provide as with classes for source object.

→ Listner: It is known as event handler. Responsible for generating response to an event. In Java, listner is also an object. Listner waits unfill it receives an event. Once the event is received, the listner processes the event and then returns.

steps:

→ The user clicks the button and the event is generated
→ Now the obj. of concerned event class is created automatically and information about the source & the event get populated with in some object.
→ Event object is forwarded to the method of registered listner class.
→ The method is now get executed and returns.

Date
Page

7. Differentiate event source & event listner? Describe different event classes.

=)

Event source:

The source is an object on which event occurs. Source is responsible for providing information of the occured event to it's handler. Java provide as with classes for source object.

Event listner:

It is also known as event handler. Listner is responsible for generating response to an event. From java implementation point of view, the listner is also an object. Listner waits until it receives an event. One the event is received, the listner process the event and then returns.

Event classes.

(a) ActionEvent → Generated when a button is pressed, a list item is double-clicked, or a menu item is selected.

(b) AdjustmentEvent → Generated when a scroll bar is manipulated.

(c) ComponentEvent → Generated when a component is hidden, moved, resized, or become visible.

(d) ContainerEvent → Generated when a component is added or removed from a container.

(e) FocusEvent → Generated when a component gains or loses keyboard focus.

(f) InputEvent → Abstract superclass for all component input event classes.

(g) ItemEvent → Generated when a check box or list item is checked; also occurs when a choice selection is made or a checkable menu item is selected or deselected.

(h) KeyEvent → Generated when input is received from the keyboard.

(i) MouseEvent → Generated when the mouse is dragged, moved, clicked, pressed or released; also generated when the mouse enters or exists a component.

(j) MouseWheelEvent → Generated when the mouse wheel is moved.

(k) TextEvent → Generated when the value of a text area or text field is changed.

(l) WindowEvent → Generated when a window is activated, closed, deactivated, deiconified, iconified, opened or quit.

8. How ActionEvent is generated? Explain JButton event handling with an example.

=) An ActionEvent is generated when a button is pressed, a list item is double-clicked, or a menu item is selected.
It contains
String getActionCommand(), int getModifiers() and
long getWhen() methods.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class HandleJButton extends JFrame implements
ActionListner {
    JLabel lbl;
    public HandleJButton() {
        JButton btn1 = new JButton("Yes");
        JButton btn2 = new JButton("No");
        lbl = new JLabel("");
        btn1.addActionListner(this);
        btn2.addActionListner(this);

        add(lbl);
        add(btn1);
        add(btn2);
        setSize(400, 400);
        setVisible(true);
        setLayout(new FlowLayout());
    }
```

```
@Override
public void ActionPerformed (ActionEvent ae) {
    String action = ae.getActionCommand();
    if (action.equals("Yes")) {
        lbl.setText("Yes Button pressed");
    }
    else if (action.equals("No")) {
        lbl.setText("No Button pressed");
    }
}
public static void main (String[] args) {
    new HandleJButton();
}
}
```

Here Two buttons [Yes] and [No] are created and a lebel is created. whenever user click yes. "Yes Button pressed" message is display in label and when click No button "No Button pressed" message is displayed.

Date
Page

9. Which event is generated when you click mouse? Explain mouse event handling mechanism with suitable programming code in Java.

→ When I click mouse, MouseEvent is generated.

Mouse event handling :-

```
import java.awt.*;
import java.awt.event.*;
public class MouseEvents extend Frame implements
MouseListener, MouseMotionListner {
    Label l;
    int mouseX, mouseY;
    MouseEvents() {
        setTitle("Mouse Demo");
        mouseX = 0;
        mouseY = 0;
        addMouseListner(this);
        addMouseMotionListner(this);
        l = new Label();
        l.setBounds(20,50,400,200);
        add(l);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
```

```
public void MouseClicked (MouseEvent me){
    l.setText("mouse clicked");
}
public void mouseEntered (MouseEvent me){
    l.setText("mouse entered");
}
public void mouseExited (MouseEvent me){
    l.setText("mouse exited");
}
public void MousePressed (MouseEvent me){
    l.setText("Mouse pressed");
}
public void mouseReleased (MouseEvent me){
    l.setText("mouse released");
}
public void mouseDragged (MouseEvent me){
    mousex = me.getX();
    mousey = me.getY();
    l.setText("Dragging mouse at "+mousex+", "+
        mousey);
}
public void mouseMoved (MouseEvent me){
    l.setText("moving mouse at "+me.getX()+", "+
        me.getY());
}
public static void main (String[] args){
    new MouseEvents();
}
}
```

Date
Page

10. What are the sources of an event? Describe different event listner interface of java.awt.event packages.

⇒ Sources of an event:

(a) Button → action events when button is pressed.

(b) Check Box → item events when the check box is selected or deselected.

(c) Choice → item events when the choice is changed.

(d) List → action event when an item is double clicked, item event when an item is selected or deselected.

(e) Menu item → action event, when a menu item is selected; item events, when a checkable menu item is selected or deselected.

(f) scroll bar → adjustment events when the scroll bar is manipulated.

(g) Text components → text events when the user enters a character.

(h) Window → window events when a window is activated, closed, deactivated, deiconified, iconified, opened or quit.

Event listner interfaces:-

(a) ActionListner → defines one method to receive action events.

(b) AdjustmentListner → defines one method to receive adjustment events.

(c) componantListner → defines four methodes to recognize when a componant is hiden, moved, resized or shown.

(d) ContainerListner → defines two methods to recognize when a component is added to or removed from a container.

(e) FocusListner → defines two methods to recognize when a component gain or loses keyboard focus.

(f) ItemListner → defines one method to recognize when the state of an item changes.

(g) KeyListner → defines three methods to recognize when a key is pressed, released, or typed.

(h) MouseListner → define five methods to recognize when the mouse is clicked, enters a component, exits a component, is pressed or is released.

(i) MouseMotionListner → defines two methods to recognize when the mouse is dragged or moved.

(j) MouseWheelListner → defines one method to recognize when the mouse wheel is moved.

(k) TextListner → defines one method to recognize when a text value changes.

(l) WindowFocusListner → defines one method to recognize when a window gain or loses input focus.

(m) WindowListner → defines seven methods to recognize when a window is activated, closed, deactivated, deiconified, iconified, opened or quit.

11. How swing is differ from AWT ? What are the advantages of swing ?

=>

AWT	swing
→ AWT components are platform dependent.	→ Java swing components are platform independent.
→ AWT components are heavy weight.	→ swing components are light weight.
→ AWT doesn't support pluggable look and feel.	→ swing supports pluggable look and feel.
→ AWT provides less components than swing.	→ swing provides more powerful components such as table, lists, scrollpanes, colorchoo tabbedpane etc.
→ AWT doesn't follow MVC where model represents data, view represents presentation and controller acts as an interface between model and view.	→ swing follows MVC.

Advantages of swing :-

→ provides both additional components and added functionality to AWT - replacement component.

→ swing components can change their appearance based on the current look and feel library that's being used.

→ Follows Model + View - Controller (MVC) and can provide much more flexible UI.

→ swing provides extras for components such as
- Icons on many components
- Decorative borders for components
- Tooltips for components

⇒ components are lightweight

→ provides built-in buffering

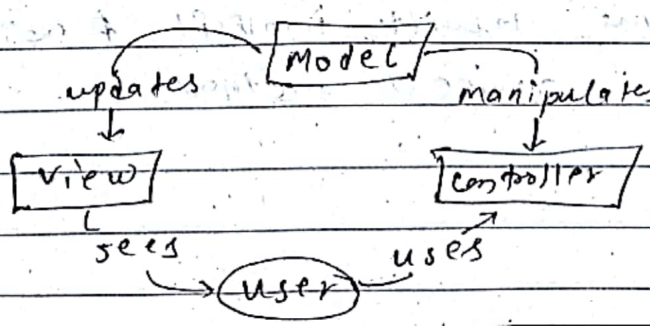
12. Explain the mechanism of separating application code with GUI in swing.

⇒ Mechanism of separating application code with GUI is called model-view-controller (MVC). MVC is a well known software architecture pattern idea to implement user interfaces on computers by dividing an application in to three parts. Main goal of MVC is to seperate internal representations of an application from the ways information are presented to the user.
MVC patterns has following 3 components:-

① Model that manages data, logic and rules of the application

② View that is used to present data to user.

③ Controller that accepts i/p from the user and converts it to commands for the Model & view.



Date
Page

13. What is Java servlet? How do you differentiate ja applet, java servlet and java server pages?

⇒ Java servlet is a technology that is used to create w application.

① Java applet
- It is a small application which is written in java an delivered to user in the form of bytecode.
- Applets are executed on client side
- Applets are used to provide interactive features to web applications that can not be provided by HTML alone like capture mouse input etc.
- Its lifecycle init(), stop(), paint(), start(), dest
- Applets are two types
  Untrusted applets and trusted applets.

② Java servlet
- servlet is a Java programming language class used to extend the capabilities of a server.
- servlets are executed on server side.
- lifecycle, init(), service() and destroy().
- It is of two types
  Generic servlet and HTTP servlet
- It does not have in-built implicit objects.
- we can override service() method.

Date
Page

(5) Java server page

- JSP is a HTML based code
- Jsp is slower than servlet because the first step in the JSP lifecycle is the translation of JSP to java code then compile.
- Jsp only accepts HTTP requests.
- In JSP session management is automatically enabled.
- There are inbuilt implicit objects.

19. Why servlet is important in Java? What are the advantage of using servlet in your application?

=> servlet is an important technology in Java for web application. It introduce internet in Java. It is
- an API that provides many interfaces and classes including documentations.
- an interface that must be implemented for creating any servlet.
- a class that extends the capabilities of the servers and responds to the incomming requests. It can respond to any type of requests.
- a web component that is deployed on the server to crate dynamic web page.

Date
Page

Advantages of using servlets:-

- Performance is significantly better. Servlets execute within the address space of web server. It is not necessary to create a separate process to handle each client request.

- servlets are platform-independent because they are written in Java.

- The Java security manager on the server enforces a set of restrictions to protect the resources on a server machine.

- The full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases or other software via the sockets and RMI mechanisms.

Date
Page

15. How multiple clients are handled in Java servlet? Describe its life cycle.

=) In Java servlet, each request is processed in a separated thread. There is a pool of threads to process requests. A single servlet can create multiple threads and each threads uses separate instance of servlet.

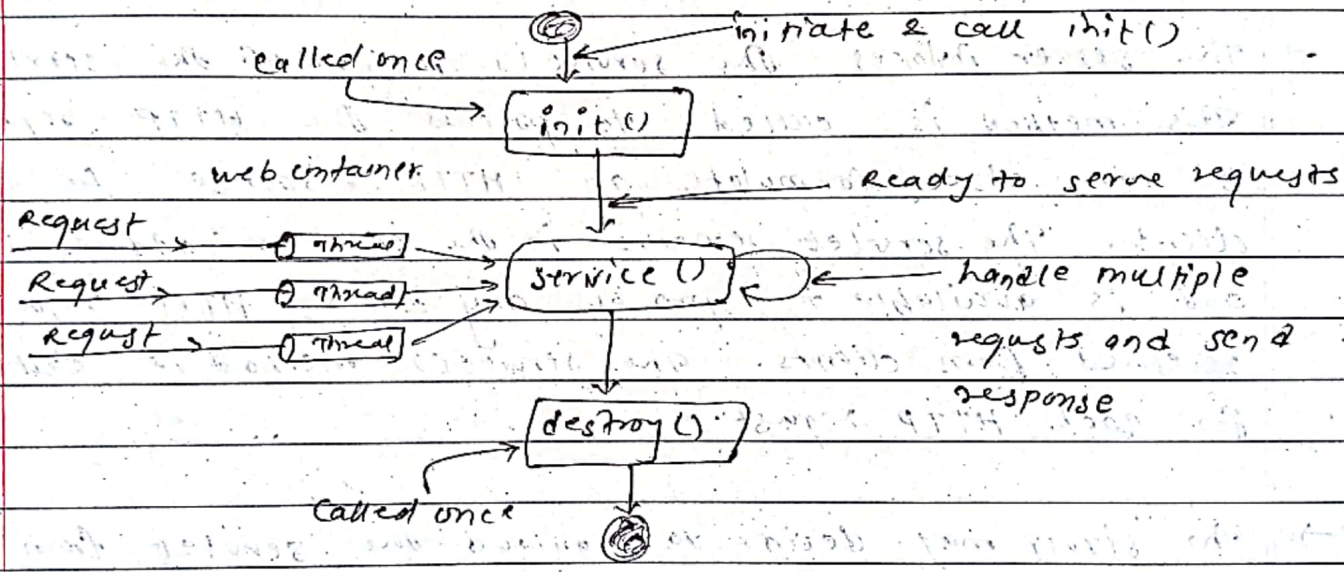


fig. servlet life cycle.

Three methods, init(), service() and destroy() are central to the life cycle of a servlet.

→ Assume that a user enters a URL to a web browser. Browser generates a HTTP request for this URL. Request is then sent to the appropriate server.

→ The web server receives this HTTP request. The server maps this request to a particular servlet. The servlet is dynamically retrieved and loaded into the address space of the server.

→ The server invokes the init() method of the servlet. This method is invoked only when the servlet is first loaded into memory. It is possible to pass initialization parameters to the servlet so it may configure itself.

→ The server invokes the service() method of the servlet. This method is called to process the HTTP request. It may also formulate an HTTP response for the client. The servlet remains in the servers address space and is available to process any other HTTP requests received from clients. The service() method is called for each HTTP request.

→ The server may decide to unload the servlet from its memory. The algorithms by which this determination is made are specific to each server. The server calls the destroy method to relinquish any resources such as file handles that are allocated for the servlet. Important data may be saved to a persistent store. The memory allocated for the servlet and its objects can then be garbage collected.

Date
Page

16. What are the necessary environment and interfaces required to develop and run java servlet. Explain with the help of simple servlet.

⇒ necessary environments :-

- Download and install java
- Download and configure Apache Tomcat web server
- Integrated development environment setup.
- Finally test setup ie: verification of installation.

servlet interface method:

→ void destroy() : called by the servlet container at the end of servlet life cycle.

→ void init (ServletConfig config) : When servlet container starts up it loads all the servlets and instantiates them.

→ void service (ServletRequest req, ServletResponse res) : This is the only method that is called multiple times during servlet life cycle.

→ ServletConfig getServletConfig() : Returns a ServletConfig object, which contains initialization and startup parameters for this servlet.

→ java.lang.String getServletInfo() : Returns information about the servlet, such as author, version and copyright.

eg.

index.html

```
<a href = "welcome"> click here to call the servlet </a>
```

DemoServlet.java

```
import java.io.*;
import javax.servlet.*;
public class DemoServlet implements servlet {
    ServletConfig config = null;
    public void init(ServletConfig config) {
        this.config = config;
        System.out.println("Initialization complete");
    }
    public void service(ServletRequest req, ServletResponse
    res) throws IOException, ServletException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.print("<html>");
        out.print("<body>");
        out.print("<h1> servlet Example Program </h1>");
        out.print("</body>");
        out.print("</html>");
    }
    public void destroy() {
        s.o.p("servlet life cycle finished");
    }
}
```

```
    public ServletConfig getServletConfig() {
        return config;
    }
    public String getServletInfo() {
        return "A demo program";
    }
}
```

web.xml

```
<web-app>
<servlet>
    <servlet-name> NewServlet </servlet-name>
    <servlet-class> DemoServlet </servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name> NewServlet </servlet-name>
    <url-pattern> /welcome </url-pattern>
<servlet-mapping>
<web-app>
```

Date
Page

17. Why do we need: Servlet, ServletRequest and ServletResp interfaces while developing java servlet? List the methods defined by each of the above interfaces.

=) Each servlet must implements .. servlet interface. It decla init(), service() & destroy() methods that are called by the server during life cycle of a servlet.
ServletRequest interface enables a servlet to obtain information about a client request.
The ServletResponse interface enables a servlet to formulate a response for client.

Methods defined by interfaces

1. Servlet interface
   - getServletConfig()
   - getServletInfo()

2. ServletRequest interface
   - Object getAttribute(String attr)
   - String getCharacterEncoding()
   - int getContentLength()
   - String getContentType()
   - ServletInputStream getInputStream() throws IOExcep
   - String getParameter(String pname)
   - Enumeration getParameterNames()
   - String[] getParameterValues(String name)
   - String getProtocol()
   - BufferedReader getReader() throws IOException

- String getRemoteAddr()
- String getRemoteHost()
- String getScheme()
- String getServerName()
- int getServerPort()

8. ServletResponse interface
- String getCharacterEncoding()
- ServletOutputStream getOutputStream() throws IOException
- PrintWriter getWriter() throws IOException
- void setContentLength(int size)
- void setContentType(String type)

18. Write a java program to demonstrate the way to read parameters from servlet.

main.html

```
<form action = "NewServlet" method = "post">
    <input type = "text" name = "text1">
    <input type = "text" name = "text2">
    <input type = "submit" value = "ok">
</form>
```

Date
Page

NewServlet.java

```
@WebServlet (name = "NewServlet", urlPatterns = {"/NewServlet"})
public class NewServlet extends HttpServlet {
    @Override
    protected void doPost (HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {
        response.setContentType ("text/html");
        PrintWriter out = response.getWriter();
        String t1 = request.getParameter("text1");
        String t2 = request.getParameter ("text2");

        out.println ("<h1> Welcome " + t1 + "</h1>");
        out.println ("<h2> You are from " + t2 + "</h2>");
        out.close();
    }
}
```

19. What is filter in servlet? Write a simple servlet named as AddServlet with the url pattern /addServlet which gets the two request parameter (num1 and num2) from jsp page and add those value and redirect along with summed value to another servlet SquareServlet with url pattern /sqservlet, which square the summed value and send response to the client.

=> A filter is an object that is used throughout the pre- and post-processing stages of a request. Filters are mostly used for filtering tasks such as server-side logging, authentication, authorization, input validation and so on.

isptoserv.jsp

```
<%@ page language = "java" %>
<%
    request.setAttribute ("num1", 2);
    request.setAttribute ("num2", 4);
    String strViewPage = "../addServlet";
    RequestDispatcher dispatcher = request.getRequestDispatcher
    (strViewPage);
    if (dispatcher != null){
        dispatcher.forward (request, response);
    }
%>
```

Date
Page

AddServlet.java

```
@WebServlet (name = "AddServlet", urlPatterns = {"/addServl
public class AddServlet extends HttpServlet {
    @Override
    public void doGet (HttpServletRequest req, HttpServlet-
    Response res) throws IOException, ServletException {
    int num1 = req.getParameter ("num1");
    int num2 = req.getParameter ("num2");
    res.setContentType ("text/html"); int num = num1+num2
    PrintWriter out = res.getWriter ();
    out.print("<a href = "sqservlet?num = "num" >
        go </a>
    out.close ();
    }
}
```

SquareServlet.java

```
@WebServlet (name = "SquareServlet", urlPatterns = {"/
    sqservlet"})
public class SquareServlet extends HttpServlet {
    @Override
    public void doGet (HttpServletRequest req, HttpServletResp
    res) throws IOException, ServletException {
    res.setContentType ("text/html");
    PrintWriter out = res.getWriter ();
    String n = req.getParameter ("num");
    int n = String.valueOf (n); Integer.parseInt (n);
    n = n*n;
    out.print ("square of sum =" + n);
    out.close ();
} }
```

Date
Page

90. How do you read and write session info in java servlet? What are the methods defined by HttpSession interface.

⇒ session can be created via getSession() method of HttpServletRequest. Httpsession object is returned. This object can store a set of bindings that associate names with objects.
It have following methods for bind binding
→ setAttribute().
→ getAttribute()
→ getAttributeNames()
→ removeAttribute()

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DateServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServlet-
    Response res) throws ServletException, IOException {

    HttpSession hs = req.getSession(true);
    res.setContentType("text/html");
    PrintWriter pw = res.getWriter();
    Date date = (Date) hs.getAttribute("date");
    if (date != null) {
        pw.print("Last access: " + date + "<br>");
    }
```

```
        date = new Date();
        hs.setAttribute("date", date);
        pw.println("current date: " + date);
    }
}
```

Here getAttribute() use to read session info and setAttribute() use to write session info.

When we first request this servlet, the browser displays one line with current date and time info. On subsequent invocations, two lines are displayed. The first line shows the date and time when the servlet was last accessed. The second line shows current date & time.

Date
Page

21. How can you exchange information between different servlets ? Explain each with suitable section of code.

⇒ Http protocol is stateless so we need to ~~state~~ maintain state using session tracking techniques inorder to exchange information between different servlets. It has following techniques.

(1) cookies
(2) Hidden form field
(3) URL Rewriting
(4) Http session.

(1) Cookies

It is a small piece of information that is persisted between the multiple client requests. cookies is stored in client web browser.

index.html

```
<form action = "servlet1" method = "post">
    Name: <input type = "text" name = "userName" /> <br />
    <input type = "submit" value = "go" />
</form>
```

FirstServlet.java

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

```
public class FirstServlet extends HttpServlet {
  public void doPost (HttpServletRequest request, HttpServletR
  response) {
    try {
      response.setContentType ("text/html");
      PrintWriter out = response.getWriter();
      String n = request.getParameter ("userName");

      out.print (" Welcome " + n);
      Cookie ck = new Cookie ("uname", n);
      response.addCookie (ck);
      out.print ("<form action = "servlet2" >");
      out.print ("<input type = "submit" value = 'go' >");
      out.print ("</form>");
      out.close();
    }
    catch (Exception e) {
      System.out.println (e);
    }
  }
}
```

second servlet.java

```
import . . . ;
public class second servlet extends HttpServlet {
  public void doPost (HttpServletRequest request,
  HttpServletResponse response) {
    try {
```

```
response.setContentType("text/html");
PrintWriter out = response.getWriter();

Cookie ck[] = request.getCookies();
out.print("Hello " + ck[0].getValue());
out.close();
}
catch(Exception e){
    System.out.println(e);
}
}
}
```

(ii) Hidden form field.

In this case a hidden (invisible) textfield is used for maintaining the state of an user and passed to another servlet.

This is better if we have to submit form in all the pages and we don't want to depend on the browser.

changing code.

FirstServlet.java

```
out.print("<form action='servlet2'>");
out.print("<input type='hidden' name='uname'
        value='"+n+"'>");
out.print("<input type='submit' value='go'>");
```

Second servlet.java

String n = request.getParameter("uname");

(iii) URL rewritting

Here we append a token or identifier to the URL of the next servlet or the next resource. We can send parameter name/value pairs using

url?name1=value1&name2=value2&??

It works whether cookie disable or enable and extra form submission is not required.

First servlet.java

// appending the username in the query string

out.print("visit");

second servlet.java

// getting value from the query string

String n = request.getParameter("uname");

(iv) HTTP session

In this case container creates a session id. for each user. Container uses this id to identify the particular user. Httpsession can be used to perform two tasks:

(i) bind objects
(ii) view and manipulate info about a sessin such as session identifier, creation time, last accessed time.

↳ public Httpsession getsession() : Returns the current session associated with this req. or if the req. doesn't have a session one.

↳ public Httpsession getsession(boolean create) : Returns the current Httpsession asso. with this req. or if there is no current sess. and create is true, returns a new session.

Firstservlet.java

```
Httpsession session = request.getsession();
session.setAttribute("uname", n);
out.print("<a href='servlet2'>visit</a>");
```

Secondservlet.java

```
Httpsession session = request.getsession(false);
String n = (String) session.getAttribute("uname");
```

22. What are the difference between get and post method ? Explain the usages of cookies.

=>

The get method sends the encoded user information appended to the page request. The page and encoded info are separated by ? symbol as :

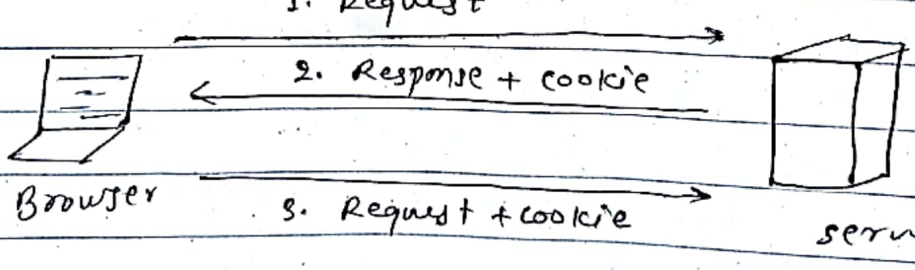
'http://test.com/hello?key1=value1&key2=value2

post method packages the info in exactly the same way as get method, but instead of sending it as a text string after a ? in the url it sends it as a separate message.

This message comes to the backend program in the form of the standard input which can parse & use for processing.

doGet() is used to recieve get request and doPost() for post method in servlet.

cookie is a small piece of information that is persisted between the multiple client request. A cookie has a name, a single value, and optional attributs such as comment, path and domain, max age and version number.



Http is stateless protocol, so by default each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by user, cookie is added with request by default. Thus we recognize the user as the old user. In such way cookies are used to track the state.

13. What are the advantages of JSP? Explain different standard tag library of JSP.

=> Advantages of JSP:-

(i) Extension to servlet:
We can use all the features of the servlet in JSP. In addition we can use implicit objects, predefined tags, expression language and custom tag in JSP, that makes easy.

(ii) Easy to maintain:
JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet we mix our business logic with the presentation logic.

(iii) Fast development:
No need to recompile and redeploy if JSP page is modified. The servlet code needs to be updated

and recompiled. If we have to change the look and feel of the application.

(iv) Less code than servlet:

In JSP, we can use many tags such as action tags, JSTL, custom tags etc. that reduces the code.

standard tag library of JSP also known as (JSTL). is a collection of useful JSP tags which encapsulates core functionality common to many JSP applications.

JSTL tags can be classified, according to their functions into following.

- → Core tags
- → Formatting tags
- → SQL tags
- → XML tags
- → JSTL functions.

(i) Core tags

It provides variable support, URL management, flow control etc.

Syntax:

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core"
    prefix = "c" %>
```

It have following tags:

c:out, c:import, c:set, c:remove, c:if, c:forEach, c:url, c:redirect, c:param, c:forTokens, c:choose, c:when, c:otherwise, c:catch.

(ii) Functional tags

JSTL functions provides a number of standard functions, most of these functions are common string manipulation functions.

syntax:

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fun" %>

JSTL functional tags:

fn: contains(),	fn: trim();	fn: substring()
fn: containsIgnoreCase(),	fn: startsWith()	fn: substringAfter()
fn: endsWith(),	fn: split()	fn: substringBefore()
fn: escapeXml(),	fn: toLowerCase()	fn: length()
fn: indexOf(),	fn: toUpperCase()	fn: replace()

(iii) Formatting tags

Formatting tags provides support for message formetting num. and date formatting etc. These are used for internationalized web sites to display and format text, time, date & numbers.

syntax for including.

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

fmt: parseNumber	fmt: setBundle
fmt: timeZone	fmt: message
fmt: formatNumber	fmt: formatDate
fmt: parseDate	
fmt: bundle	
fmt: setTimeZone	

(iv) XML tags

Used for providing a JSP-centric way of manipulating & creating XML documents. XML tags provide flow control, transformation etc.

<%@ taglib uri = "http://java.sun.com/jsp/jstl/xml" prefix = "x" %>

t: out	x: otherwise
x: parse	x: if
x: set	t: transform
x: choose	x: param
x: when	

(v) SQL tags

provides SQL support. It allows the tag to interacts with RDBMS such as Microsoft SQL server, mySQL, Oracle.

<%@ taglib uri = "http://java.sun.com/jsp/jstl/sql" prefix = "sql" %>

sql: setDataSource
sql: query
sql: update
sql: param
sql: dateParam
sql: transaction

Q. Give the HTML to create a form with two elements: a textbox named firstName that holds a maximum of 50 characters, and a submit button. The form should submit its data to a JSP program called processName.jsp using the post method. Also write the processName.jsp to handle these request.

=> index.html

```
<html>
    <form action = "processName.jsp" method = "post">
Name: <input type = "text" max-size = "50" name = "firstName">
        <input type = "submit" value = "send">
    </form>
</html>
```

processName.jsp

```
<html>
    <p> hello, <% = request.getParameter ("firstName") %>
    </p>
</html>
```

25. Identify the following JSP tags:
<% %>, <%@ %>, <%! %>, <%= %>.
Also explain them.

⇒
(i) Scriptlets <% %>
↳ used to embed java code in JSP pages.
code should comply with syntactical and semantic construct of java.

```
<% int x = 5
   out.println("x = " + x);
%>
```

(ii) Declaration <%! %>
→ Declarations are used to define methods and instance variables. Do not produce any output that is sent to client.

```
<%!
   int myVar = 123;
%>
```

(iii) Expression <%= %>
→ used to write dynamic content back to the browser.
→ If the o/p of expression is Java primitive the value is printed back to the browser.
→ If the o/p is an object then the result of calling toString on the object is output to the browser

eg.

<% = Math.squart(100) %>

(iv) Include directive <%@ %>

→ used to insert template text and JSP code during the translation phase.

→ content of the included file specified by the directive is included in the including JSP page.

eg. <%@ include file = "included.jsp" %>

26. Explain the mechanisms of conversion from JSP to servlet with reference to life cycle method of servlet & JSP.

⇒

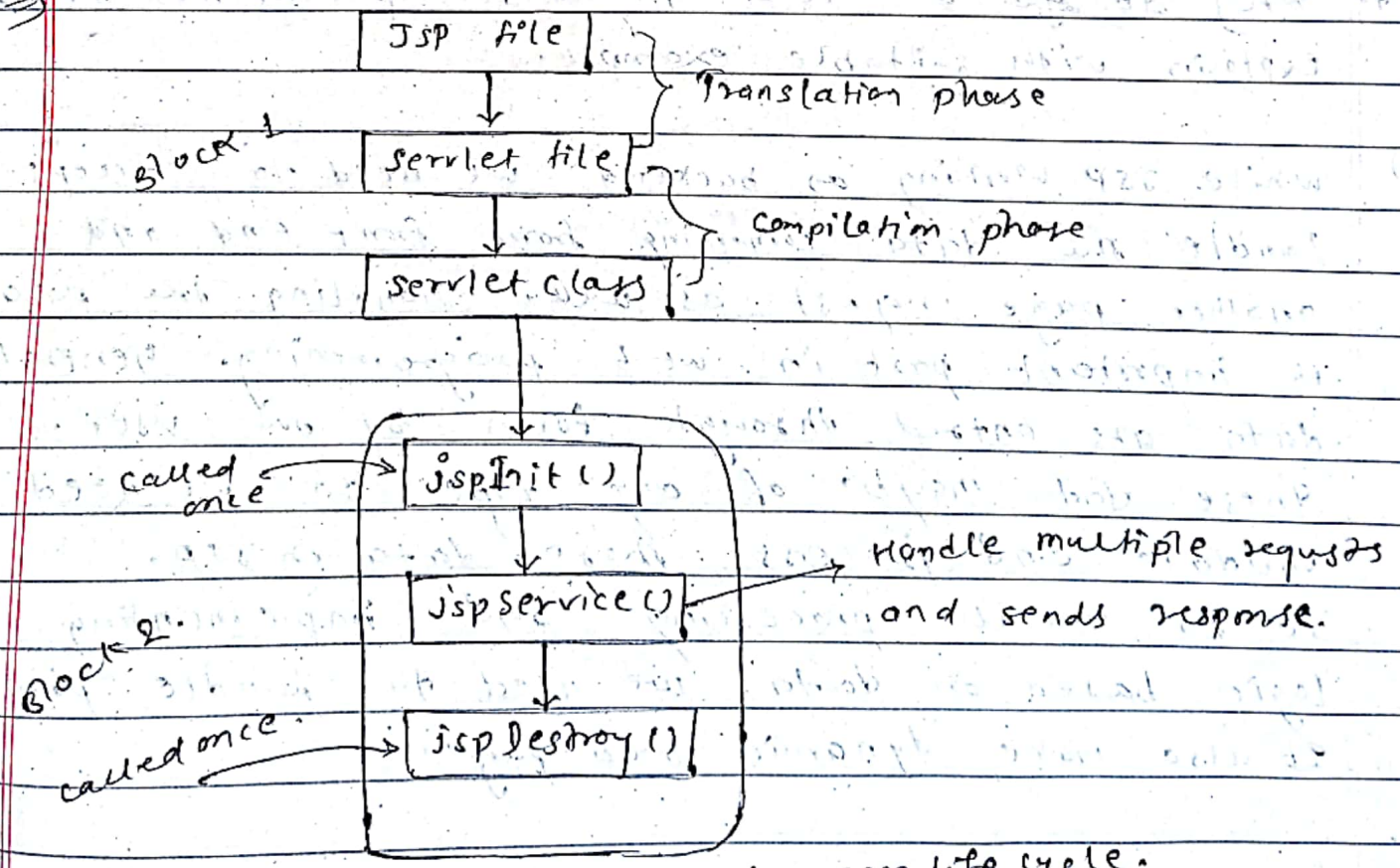


fig. JSP life cycle.

Date
Page

In the lifecycle of JSP Block 2 is similar to the lifecycle of servlet. That means ultimately JSP is converted into servlet.
First JSP file is translated into servlet file on the way. Then the file is compiled through servlet engine. After that similar process to servlet happens.

The following are the paths followed by JSP:
→ Compilation
→ Initialization
→ Execution
→ Cleanup

27. Why do ~~go~~ we need to handle parameters in JSP? Explain with suitable example.

⇒ While JSP working as backend we need to accept and handle the data comming from front end and another page request as well. Handling the parameter is important part in web programming. Generally data are entred through form by the user. These data maybe of any type so we need to validate and process these data in JSP.
So for overall processing and implementing business logic based on data we need to handle parameters. It also make dynamic web page.

Date
Page

simple example of handling parameter comming from html page is shown below:

index.html

```
<html>
 <body>
    <form action = "NewFile.jsp" method = "POST">
     Parameter 1 : <input type = "text" name = "p1" >
     <br />
     Parameter 2 : <input type = "text" name = "p2" >
     <br />
     <input type = "submit" value = " Enter values" />
    </form>
 </body>
</html>
```

NewFile.jsp

```
<html>
<head>
<title> Using ~~post~~ getParameter() to read data from page
 </title>
</head>
<body>
 <h1> get parameter() function usage..... </h1>
 <ul>
 <li><p><b> Parameter 1 : </b>
  <%= request.getParameter ("p1") %>
 </p></li>
```

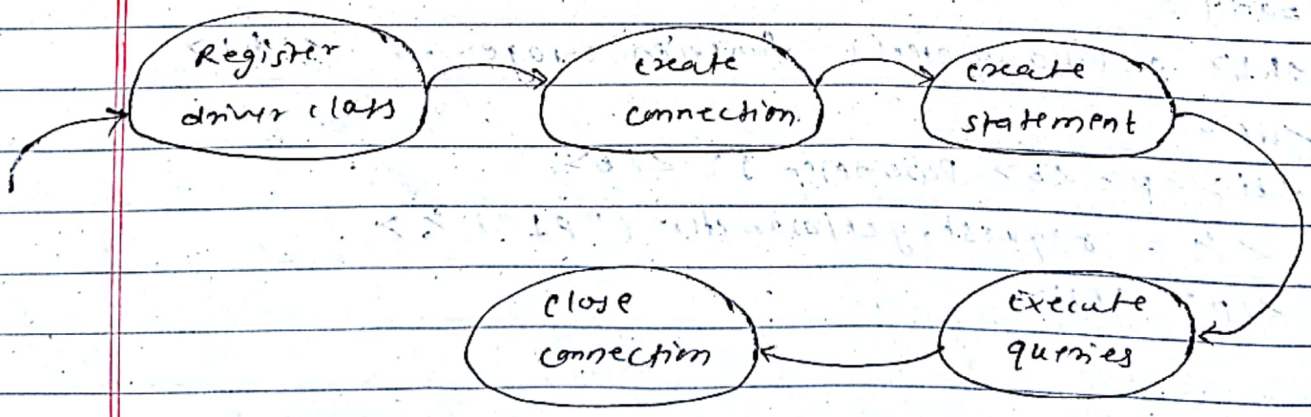
```
<li> <p> <b> Parameter 2 : </b>
    <%= request.getParameter("p2") %>
</p>
</li>
</ul>
</body>
</html>
```

28. What is JDBC ? Explain how Java program access the database ?

⇒ JDBC is a Java Database connectivity.

- It provides a standard library for Java programs to connect to a database and send it commands using SQL.
- It generalizes common database access functions into a set of common classes and methods.
- It enables us to write Java program to access any kind of tabular data stored in a Relational Database.
- It is a part of JAVA SE.

steps for accesing database.



Date
Page

DB from

To access then Java program :-

→ Import the necessary class
→ Load JDBC driver
→ Identify the data source (conn URL)
→ establish conn.
→ create stmt obj
→ Execute query str using stmt obj
→ Retrieve data from returned Resultset object
→ close Resultset & stmt & conn. obj in order.

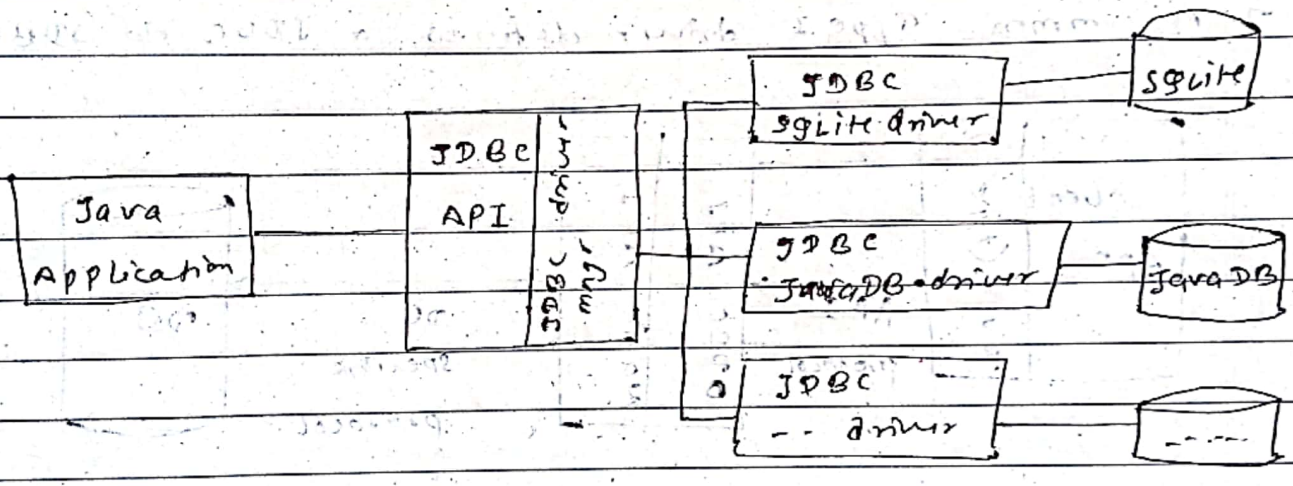


fig. Java database connectivity.

step 1 : connect
step 2 : Query / update DB
step 3 : Recieve Result

Date
Page

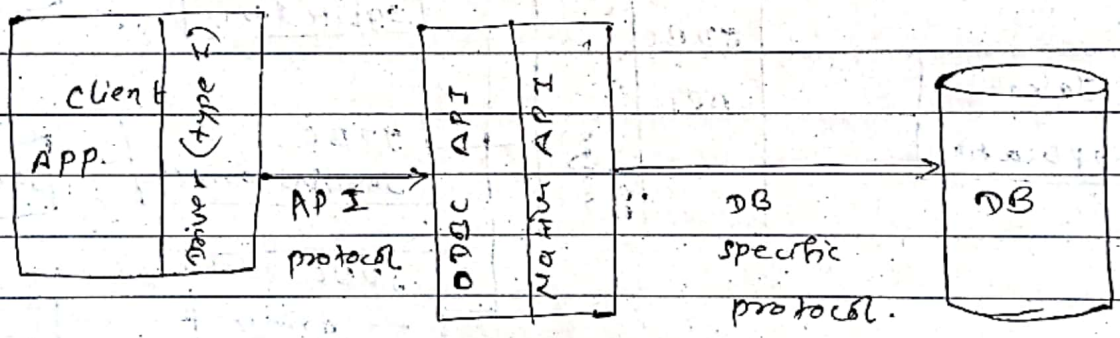
29. What are the types of JDBC drivers? Explain the usages of JDBC driver.

⇒ There are four types of JDBC drivers.
Type I, II, III & IV.

(i) Type I

→ It provides mapping between JDBC and access API of a database. Access API calls the native API of the DB to establish comm°.

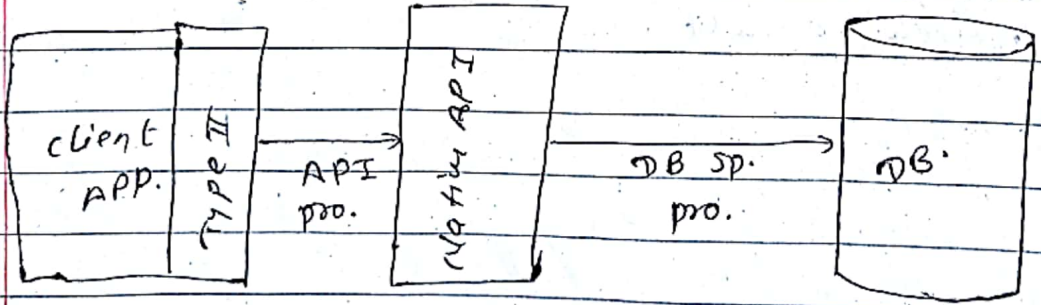
→ A common Type I driver defines a JDBC to ODBC bridge.



(ii) Type II

→ It communicates directly with native API.

→ Driver converts JDBC method calls into native calls of DB.

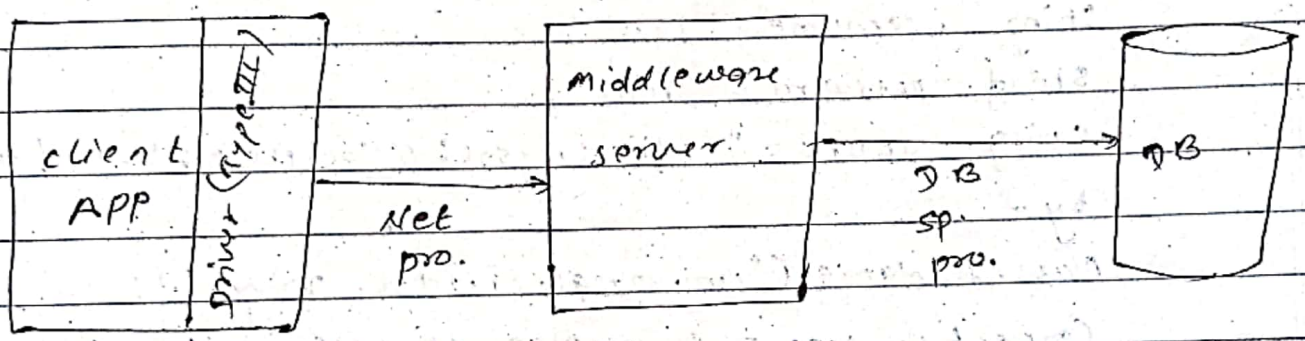


(iii) Type III

→ It makes calls to a middleware component running on another server.

→ This comm^n uses a dB independent net protocol.

→ middle-tier may use Type I or II JDBC driver to comm^n. with DB.

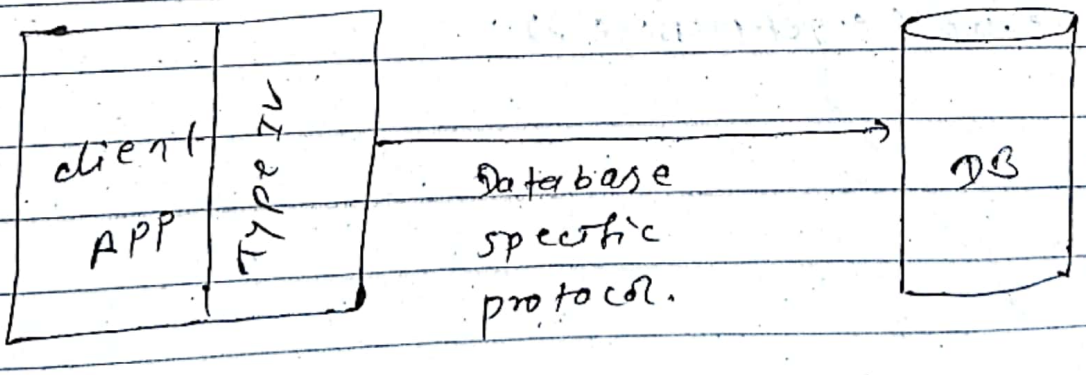


(iv) Type IV

→ It is an all-Java driver. That is also called a thin driver.

→ It issues requests directly to the DB using native protocol.

→ It can be used directly on platform with a JVM



Date
Page

80. Write a JDBC program to input the info of a ~~particular~~ product [id, name, price, qty] and in a table "tbl-product". (Assume DB name yourself)

=>
```
import java.sql.*;
class SqlDemo{
  public static void main (String[] args) {
      String username = "root";
      String password = "root";
      String dbUrl = "jdbc:mysql://localhost:3306/testdb";
      try{
      Class.forName("com.mysql.cj.jdbc.Driver");
      Connection conn = DriverManager.getConnection (dbUrl,
       username, password);
       Statement stmt = conn.createStatement();
       String query = "INSERT INTO tb-product VALUES (
          2, "Tea", 200, 2);
          stmt.executeQuery (query);
        conn.close();
      }
      catch (SQLException e) {
          s.o.p (e.getMessage());
      }
  }
}
```

Date
Page

31. A database "testdb" contains a table "employee" with some records having id, name, post, salary. Write a java program to display only those employees whose salary is more than 45000.

=>
```
import java.sql.*;
class SqlDemo {
    P.S.V.M (String [] args) {
        String uname = "root";
        String pwd = "admin";
        String dbUrl = "jdbc:mysql://localhost:3306/testdb";
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection conn = DriverManager.getConnection (dbUrl,
            uname, pwd);
            Statement stmt = conn.createStatement();
            String query = "SELECT * FROM employee WHERE
            salary > 45000";
            ResultSet r = stmt.executeQuery (query);
            while (r.next()){
                S.O.P (r.getInt(1) + " " + r.getString(2) + " "+
                r.getString(3) + " " + r.getDouble(4));
            }
            conn.close();
        } catch (Exception e) {
            sop (e.getMessage());
        }
    }
}
```

3.15

Date: Page

32. What are the difference between JDBC and ODBC

JDBC	ODBC
→ JDBC is object oriented	→ ODBC is procedure oriented.
→ It is used to provide conn^n between Java and database like oracle, sybase.	→ It is used to provide conn^n between front-end app (other than Java) and back-end like ms-access.
→ For Java applications	→ for Microsoft
→ Used by Java programmers to connect to DB.	→ Used between appns of different types.
→ Allows SQL-based DB access for EJB persistence & for direct manipulation from CORBA, IIOP & other server object.	→ With a small bridge program, we can use the JDBC interface to access ODBC accessible DB.
→ Is multi-threaded.	→ Is not multi-threaded.
→ We can do everything with JDBC that we can do with ODBC, on any platform.	→ ODBC can't be directly used with Java because it uses C interface.
→ For Java application, it is highly recommended to use JDBC because there are no performance & platform dependent problem.	→ For Java app. it is not recommended to use ODBC becoz performance will be down due to internal conversion and applications will become platform dependent.