• Checkbox

• Frame
  • Constructor: i] Frame ()
    ii] Frame (string title)
  • Methods : i] setsize ()
    ii] getsize()
    iii] setvisible
    iv] setTitle

There are two ways to create a frame in AWT
i] By extending frame class
ii] By creating object of frame class.

• AWT
i] Abstract Window Toolkit (AWT) is a set of application program interfaces (API - collection of classes and interfaces) used by Java Programmers to create graphical user interface (GUI) objects, such as buttons, scroll bars and windows.
ii] The AWT ~~class~~ is a class library that provides component for graphical user interface e.g. buttons, labels, textboxes.
iii] Java AWT is an API to develop GUI or window-based applications in java.
iv] Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system.
v] AWT is heavyweight i.e. its components uses the resources of system. The java.awt package provides classes for AWT API such as TextField, Label, TextAred, Choice, List etc.

AWT classes Hierarchy

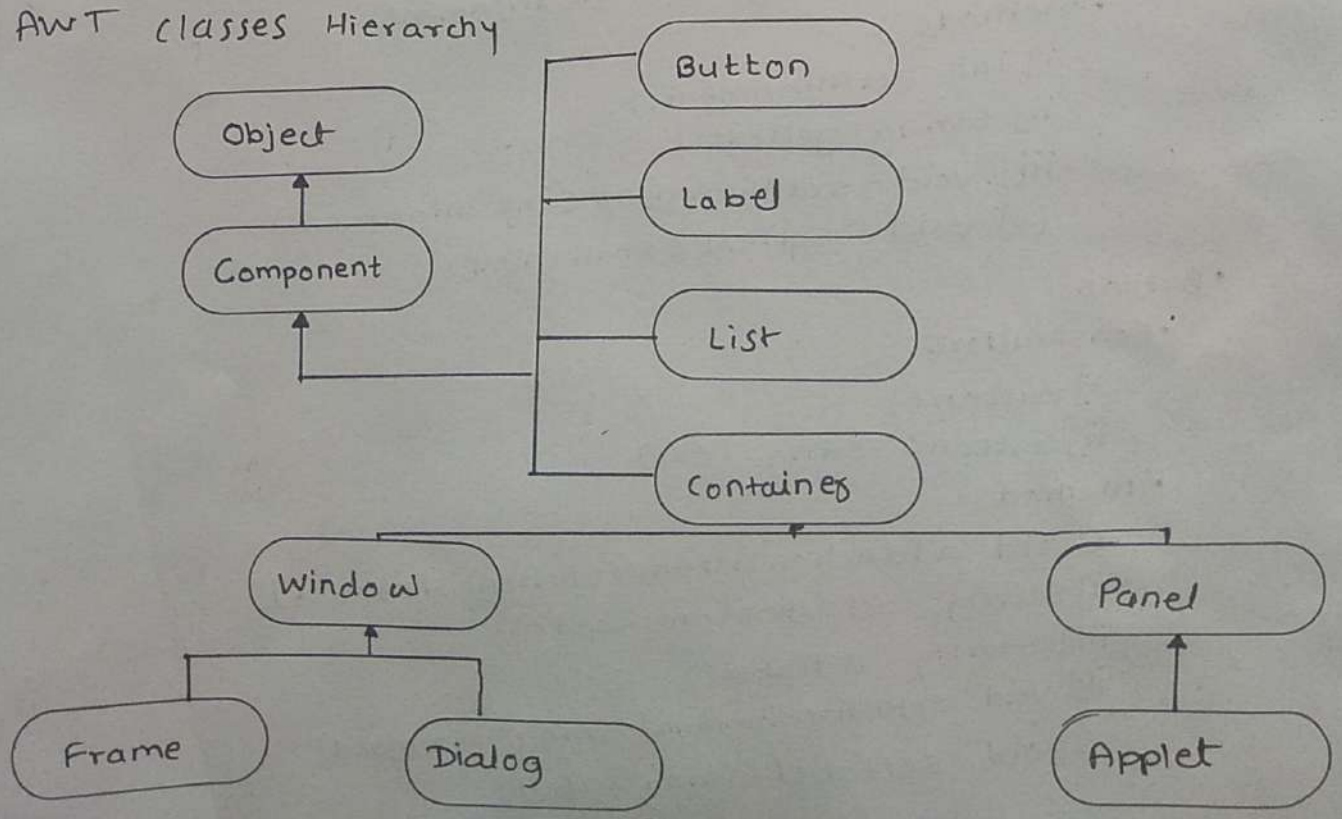


Fig. AWT classes Hierarchy

AWT Controls

i] Controls are components that allow a user to interact with your application in various ways. All AWT controls are placed inside a container and layout manager automatically positions components within a container.

ii] Thus, appearance of a window is determined by combination of the controls that in contains and layout manager used to position them.

iii] The AWT supports the following types of Controls: Labels, Push Buttons, check boxes, choice lists, Lists, Scroll bars, Text editing etc.

iv] these controls are subclass of Component class. The HeadlessException is Thrown by all AWT Controls.

- Adding and removing controls inside a container

To show control in a window, you must add it to the Frame or Panel. To do this, you must first create an instance of desired Control and then add it to a window by calling add() method, which is defined by container class

i] Component add(component compobj)
ii] void remove(component obj)

- Label
  - Constructors
    i] Label()
    ii] Label(string text)
    iii] Label(string text, int alignment)
  - Methods
    i] int getAlignment()
    ii] String getText()
    iii] void setAlignment(int alignment)
    iv] void setText(String text)

- Button
  - Constructors
    i] Button()
    ii] Button(String text)
  - Methods
    i] void addActionListener(ActionListener al)
    ii] String getActionCommand()
    iii] String getLabel()
    iv] void setActionCommand(string command)
    v] void setLabel(String Label)

- Checkbox
  - Constructors
    i] Checkbox ()
    ii] Checkbox (string label)
    iii] Checkbox (string label, boolean state)
    iv] Checkbox (string label, boolean state, CheckboxGroup group)
  - Methods
    i] void addItemListener (ItemListener l)
    ii] CheckboxGroup getCheckboxGroup()
    iii] String getLabel()
    iv] boolean getstate()
    v] void setCheckboxGroup (CheckboxGroup g)
    vi] void setLabel (String label)
    vii] void setstate (boolean state)
- CheckboxGroup

It is possible to create a set of mutually exclusive check boxes in which one and only one check box in the group can be checked at any one time. These check boxes are often called radio buttons.

  - Constructor
    i] CheckboxGroup()
  - Methods
    i] Checkbox getselectedCheckbox()
    ii] void setselectedCheckbox (Checkbox box).

——— XOX ———

Choice

- Constructor
  i] choice()
- Methods
  i] void add(String item)
  ii] void addItemListener (ItemListener obj)
  iii] String getItem(int index)
  iv) int getItemCount()
  v) int getselectedIndex()
  vi) String getselectedItem()
  vii) void insert (String item, int index)
  viii) void remove (int position)
  ix) void remove (String item)
  x) void removeAll()
  xi) void select (int pos)
  xii) void select (String str)

List

- Constructors
  - i] List()
  - ii] List(int rows)
  - iii] List(int rows, boolean multipleMode)
- Methods
  - i] void add(string item)
  - ii] void add(string item, int index)
  - iii) void addActionListener(ActionListener obj)
  - iv) void addItemListener(ItemListener obj)
  - v) void deselect(int index)
  - vi) string getItem(int index)
  - vii) int getItemCount()
  - viii) string[] getItem()
  - IX) int getSelectedIndex()
  - X) int[] getSelectedIndexes()
  - XI) String getselectedItem()
  - XII) string getselectedItems()
  - XIII) void remove(int position)
  - XIV) void remove(string item))
  - XV) void removeAll()
  - XVI) void replaceItem(string newValue, int index)
  - XVII) void select(int index)

Scrollbar

- Constants
  - i] Static int ~~Horizontal~~ HORIZONTAL
  - ii] Static int VERTICAL
- Constructors
  - i] Scrollbar()
  - ii) Scrollbar(int orientation)
  - iii] Scrollbar(int orientation, int value, int visible, int min, int max)
- Methods
  - i] void AddAdjustmentListener(AdjustmentListener obj)
  - ii] int getBlockIncrement()
  - iii] int getMaximum()
  - iv] int getMinimum()
  - v] int getUnitIncrement()
  - vi] int getValue()
  - vii] void setBlockIncrement(int v)
  - viii] void setMaximum(int newMaximum)
  - IX) void setMinimum(int newminimum)
  - X] void setUnitIncrement(int v)
  - XI] void setValue(int newv)

ADT | 1

TextField

- Constructors
  i] TextField()
  ii] TextField(int columns)
  iii] TextField(String text)
  iv] TextField(String text, int columns)
- Methods
  i] void addActionListener(ActionListener l)
  ii] int getColumns()
  iii] char getEchoChar()
  iv] void setColumns(int columns)
  v] void setEchoChar(char c)
  vi] void setText(String t)
  vii] String getSelectedText()
  viii] void select(int startIndex, int endIndex)

TextAred

- Constants
  i] static int SCROLLBARS_BOTH
  ii] static int SCROLLBARS_HORIZONTAL_ONLY
  iii] static int SCRLOLLBARS_NONE
  iv] static int SCROLLBARS_VERTICAL_ONLY
- Constructors
  i] TextArea()
  ii) TextAred(int rows, int columns)
  iii] TextAred(String text)
  iv] TextAred(String text, int rows, int columns)
  v] TextAred(String text, int rows, int columns, int scrollbars)
- Methods
  void append(String str)
  int getColumns()
  int getRows()
  void insert(String str, int pos)
  void replaceRange(String str, int start, int end)
  void setColumns(int columns)
  void setRows(int rows)

Layout Manager

A layout manager automatically arranges your controls within a window by using some type of algorithm. Sometimes device to device size information gets changed then manual placement of components become tedious. A layout manager is an instance of any class that implements LayoutManager interface. The layout manager is set by the setLayout() method

Syntax: void setLayout (LayoutManager obj)

Layout Manager Classes

i] BorderLayout
ii] FlowLayout
iii] GridLayout
iv] CardLayout
v] GridBagLayout

BorderLayout

i] BorderLayout is the default layout manager for window means for frame. The BorderLayout class implements a common layout style for top-level windows

ii] It has four narrow, fixed-width components at the edges and one large area in the centre.

iii] The four sides are referred to as north, south, east ~~west~~, and west.

iv] The middle area is called the center

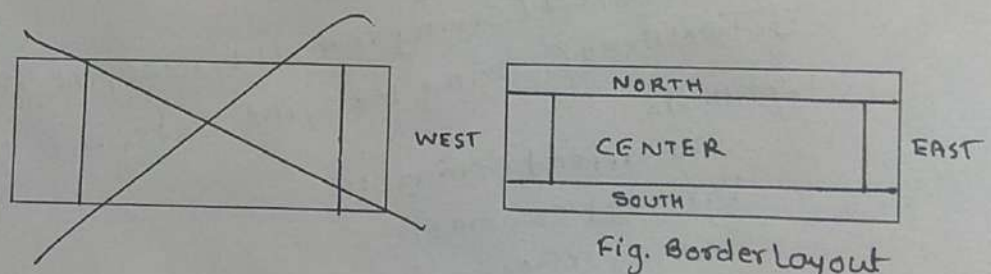


Fig. BorderLayout

- Constants
  i] static int CENTER
  ii] static int NORTH
  iii] static int SOUTH
  iv] static int EAST
  v] static int WEST
- Constructors
  i] BorderLayout()
  ii] BorderLayout (int hgap, int vgap)
- Special form of add() method for BorderLayout
  void add (Component compObj, Object region)

FlowLayout

i] FlowLayout is the default layout manager for panel means for Applet. FlowLayout implements a simple layout style, which is similar to how words flow in a text editor.

ii] Adds components from left to right, top to bottom. Components are laid out line-by-line beginning at the upper-left corner.

iii] In all cases when a line is filled layout advances to the next line.

- Constants
  - i] static int CENTER
  - ii] static int LEADING
  - iii] static int LEFT
  - iv] static int RIGHT
  - v] static int TRAILING
- Constructors
  - i] FlowLayout()
  - ii] FlowLayout(int align)
  - iii] FlowLayout(int align, int hgap, int vgap)

GridLayout

i] Gridlayout lays out components in a two dimensional grid.

ii] When you ~~inta~~ instantiate a Gridlayout, you define no. of rows and columns.

iii] The class Gridlayout arranges components in a rectangular grid.

- Constructors
  - i] Gridlayout()
  - ii] GridLayout(int rows, int cols)
  - iii] GridLayout (int rows, int cols, int hgap, int vgap)

CardLayout

i] The cardlayout class is unique among the other layout managers in that it stores several diffrent layouts.

ii] Each layout can be thought of as being on a separate index card in a deck that can be shuffled so that any card is on top at a given time

iii] This can be useful for user interfaces with optional components that can be dynamically enabled and disabled upon user input

iv] You can prepare other layouts and have them hidden, ready to be activated when needed

- Constructors
  - i] CardLayout()
  - ii] CardLayout(int hgap, int vgap
- Methods
  - i] void first(Container parent)
  - ii] void last (container Parent)
  - iii] void next (containerp parent)
  - iv) void previous (container Parent)
  - v) void show(Container parent, String name)

———— xox ————

Menu Bar

The MenuBar class provides menu bar around to a frame and is platform specific

- Constructors
  - i] MenuBar()
- Methods
  - i] Menu add(Menu m)
  - ii] Menu getMenu(int i)
  - iii] int getMenuCount()
  - iv] void remove(int index)
  - v] void remove(MenuComponent m)
  - vi] void setHelpMenu(Menu m)

———— xox ————

Dialog

i] Dialog boxes are primarily used to obtain user input and are often child windows of a top-level window

ii] Dialog boxes don't have menu bars. Dialog boxes may be modal or modeless..

iii] when a modal dialog box is active, all input is directed to it until it is closed. This means that you cannot access other parts of your program until you have closed the dialog box.

iv] when a modeless dialog box is active, input focus can be directed to another window in your program. Thus, other parts of your program remain active and accessible.

v] Dialog boxes are of type Dialog. It is used to ~~ta~~take some form of input from the user. It inherits the window class. Unlike frame, it doesn't have maximize and minimize buttons

(compObj, Object region)

FileDialog

i] Java provides a built-in dialog box that lets the user specify a file. To create a file dialog box, instantiate an object of type FileDialog.

ii] This causes a file dialog box to be displayed. Usually this is the standard file dialog box provided by the operating system. FileDialog control represents a dialog window from which user can select a file

- Constants
  i] static int LOAD
  ii] static int SAVE

SWING

i] set of API (API - set of classes and interfaces)
ii] provided to design a Graphical User Interface
iii] An extension library to AWT
iv] Includes new and improved components that enhance look and functionality of GUIs
v] Swing can be used to build standalone swing GUI Apps as well as Servlets and Applets.
vi] It employs model/view design architecture
vii] Swing is more portable and more flexible than AWT, Swing is built on top of AWT.
viii] Entirely written in Java (means do not use OS resources)

Swing Features

i] Pluggable look and feel
ii] Uses MVC architecture
iii] Lightweight components
iv] Platform Independent
v] Advance features such as JTable, JTabbedPane, JScrollPane etc.

Diffrence between AWT and SWING

ADJ/1

AWT	Swing
① AWT components are platform dependent	① Java swing components are platform-independent
② AWT components are heavyweight	② swing components are lightweight
③ AWT doesn't support pluggable look and feel.	③ Swing supports pluggable look and feel.
④ AWT provides less components than swing	④ swing provides more powerful components as tables, lists, scrollpane, tabbedpane etc
⑤ AWT doesn't follow MVC where model represents data, view represents presentation and controller acts as an interface between model and view	⑤ Swing follows MVC.

—— x o x ——

JFrame

- Constants
  - i] static int EXIT_ON_CLOSE
  - ii] static int DISPOSE_ON_CLOSE
  - iii] static int DO_NOTHING_ON_CLOSE
  - iv] static int HIDE_ON_CLOSE
- Constructors
  - i] JFrame()
  - ii] JFrame(GraphicsConfiguration gc)
  - iii] JFrame(String title)
- methods
  - i] void setDefaultCloseOperation(int operation)
  - ii] void setJMenuBar(JMenuBar menubar)
  - iii] void setLayout(LayoutManage manager
  - iv] public void add(Component c)
  - v] public void setSize(int width, int height)
  - vi] public void setVisible(boolean b)
  - vii] public void setTitle(String title)

Swing Components

i] JFrame ii] JApplet iii) JLabel iv] ImageIcon v] JTextField
vi] JTextArea vii] JPasswordField viii] JButton IX] JToggleButton
X] JCheckBox XI] JRadioButton XII] JComboBox XIII] JList
XIV] JPanel XV] JTabbedPane XVI] JScrollPane XVII] JScrollBar
XVIII] JTree XIX] JTable XX] JProgressBar XXI] JDialog
XXII] JFileChooser

The Delegation Event Model

The delegation event model is based on the Event Sources and Event Listeners. Event Listener is handler that receives the messages/events. The event Source is any object which creates the message/event. The event delegation model is based on The Event classes, The Event Listeners and Event objects.

Events: Event is an object that describes a state ~~an~~ of change in a source. Event may be generated as a consequence of a person interacting with the GUI elements such as pressing a button, entering a character through keyboard, selecting an item from a list.

Event Sources: "Source" is an object that generates an event. listeners must register with the source in order to receive event notification.

Event Listeners: A listener is an object that is notified when an event occurs. To do so 2 things should be satisfied

1) It must have been registered with one or more sources to receive notification
2) It must implement methods to receive and process these notifications.

———— x o x ————

- Process of Event Handing

i] The modern approach to handling events is based on the ~~k~~ delegation event model.
ii] In this model source generates an event and sends it to one or more listeners
iii] the listener simply wait until it receives an event
iv] Once received, the listener processes the event and then returns
v] The advantage of this design is that application logic that processes events is clearly separated from user interface that generates those events.
vi] User interface logic is able to delegate the processing of an event to a separate piece of code.
vii] In the delegation model, listeners must register with the source in order to receive event notification. each type of event has its own registration method, general form of it

```
public void addTypeListener(TypeListener obj)
```

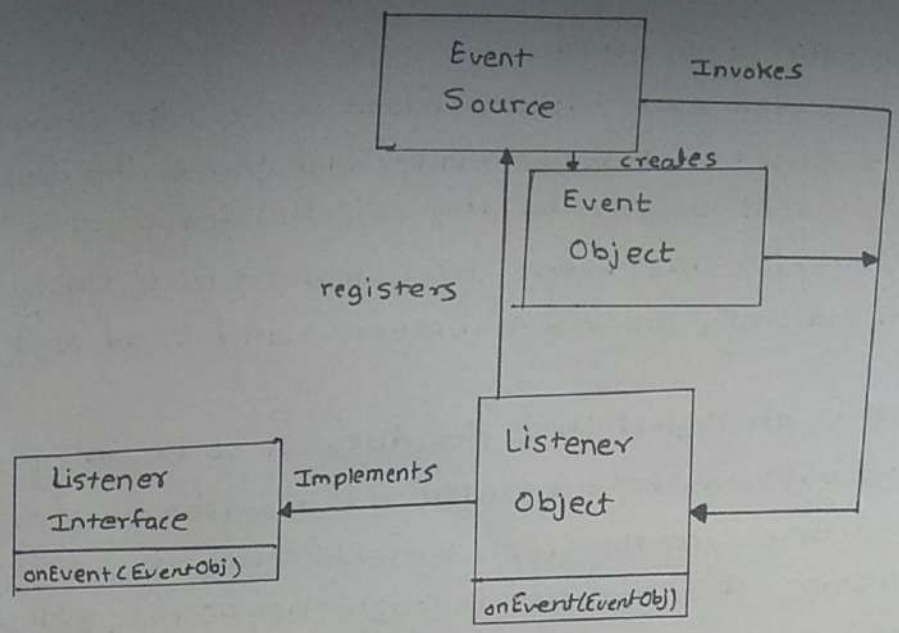


Fig. The Delegation Event Model

- Steps to handle Events
  - i] Declare and instantiate event sources (or components) such as buttons, menus, choices etc.
  - ii] Implement an interface (listener) to provide the event handler that responds to event source activity.
  - iii] Register this event handler with the event source
  - iv] Add the event source to the container like applet, frame, panel etc.
  - v] Implement the methods of listener interface and provide actions for respective type of events

——— X o X ———

- Event classes
  - i) EventObject : Super class for all events., object getSource() method
  - ii] AWTEvent : Super class for all AWTEvents
  - iii] ActionEvent: String getActionCommand() method
  - iv] FocusEvent : FOCUS_GAINED and FOCUS_LOST constants
  - v] ItemEvent : DESELECTED and SELECTED constants
  - vi] KeyEvent : KEY_PRESSED, KEY_RELEASED, KEY_TYPED ~~Event~~ constants
  - vii] MouseEvent : ~~MOUSE_EVE~~, It is subclass of InputEvent
  - viii] TextEvent : TEXT_VALUE_CHANGED constant
  - IX] WindowEvent : It is subclass of ComponentEvent
  - X] AdjustmentEvent :

Listener Interfaces

- ActionListener
  - i] void actionPerformed (ActionEvent ae)
- AdjustmentListener
  - i] void adjustmentValueChanged (AdjustmentEvent ae)
- FocusListener
  - i] void FocusGained (FocusEvent fe)
  - ii] void FocusLost (FocusEvent fe)
- ItemListener
  - i] void itemStatechanged (ItemEvent ie)
- KeyListener
  - i] void keyPressed (KeyEvent ke)
  - ii] void keyReleased (KeyEvent ke)
  - iii] void keyTyped (KeyEvent ke)
- MouseListener
  - i] void mouseClicked (MouseEvent me)
  - ii] void mouseEntered (MouseEvent me)
  - iii) void mouseExited (MouseEvent me)
  - iv] void mousePressed (MouseEvent me)
  - v] void mouseReleased (MouseEvent me)
- MouseMotionListener
  - i] void mouseDragged (MouseEvent me)
  - ii] void mouseMoved (MouseEvent me)
- TextListener
  - i] void textchanged (TextEvent te)
- WindowListener
  - i] void windowActivated (WindowEvent we)
  - ii] void windowClosed (WindowEvent we)
  - iii) void windowClosing (WindowEvent we)
  - iv) void windowDeactivated (WindowEvent we)
  - v) void windowDeiconified (WindowEvent we)
  - vi) void windowIconified (WindowEvent we)
  - vii) void windowOpened (WindowEvent we)

The ActionEvent Class

i] An ActionEvent is generated when a button is pressed, a list item is double clicked or menu item is selected.. You can obtain the command name for the invoking ActionEvent object by using the getActionCommand() method

Syntax: String getActionCommand()

ii] for eg. when a button is pressed, an action event is generated that has command name equal to the label on that button.

- Listener Interfaces

The delegation event model has two parts: sources and listeners. Listeners are created by implementing one or more of the interfaces defined by the java.awt.event package. when an event occurs, the event source invokes the appropriate method defined by the listener and provides an event object as its argument

- The ActionListener Interface

This interface defines the actionPerformed() method that is invoked when an action event occurs.

Syntax: void actionPerformed(ActionEvent ae)

- Adapter classes

Java provides a special feature, called an adapter class that can simplify the creation of event handlers in certain situations. an Adapter class provides an empty implementation of all methods in an event listener interface. Adapter classes are useful when you want to receive and process only some of the events that are handled by a particular event listener interface. You can define a new class to act as an event listener by extending one of the adapter classes and implementing only those events in which you are interested.

Computer Network

A computer Network is a group of computer systems and other computing hardware devices that are linked together through communication channels to facilitate communication and resource-sharing among a wide range of users.
Network types are LAN, MAN, WAN

IP Address

IP address stands for Internet Protocol (IP) address. An IP address is an identifier for a computer or device on a TCP/IP network. Networks using TCP/IP protocol route messages based on IP address of the destination.

IPv4 : 32 bit IP address
IPv6 : 128 bit IP address

Subnet Mask

An IP address has two components, the network address and the host address. A subnet mask seprates the IP address into the network and host address (<network> <host>). A Subnet mask is a 32 bit no. that masks an IP address and divides the IP address into network address and host address.
Subnet mask is made by setting network bits to all 1s and setting host bits to all 0's.

IP classes

class A : 1 - 126
class B : 127 - 191
class C : 192 - 223
class D : Defined for multicast operation and not used for normal op.
class E : Defined for experimental use and —||—

—xox—

Client Server Architecture

Client Server architecture is a computing model, in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to central server over a n/w or internet connection. This system shares computing resources.

Client

A client is a piece of computer hardware or software that accesses a service made available by a server. Two types: Thin client and Thick client.

Server

A computer or device on a network that manages network resources. There are many different types of servers for eg. File Server, Print server, Network Server, Database Server.

Proxy Server

A server that sits between a client application, such as web browser and a real server. It intercepts all requests to the real server ~~if~~ to see if it can fulfill the requests itself. If not it forwards the request to real server.

WWW

The World wide web (abbreviated as www, commonly known as the web) is a system of interlinked hypertext documents that are accessed via the Internet with a web browser. One can view web pages that may contain text, images, videos and other multimedia and navigate between them via hyperlinks. The documents are formatted in a markup language called HTML (HyperText Markup Language)

Difference between TCP and UDP

TCP	UDP
(i) TCP stands for transmission Control Protocol	(i) UDP stands for User Datagram Protocol
(ii) TCP is connection oriented Protocol	(ii) UDP is connectionless protocol
(iii) TCP is more reliable	(iv) UDP is not reliable
(iv) TCP is slower than UDP	(iii) UDP is faster
(vi) TCP makes checks for errors and reporting	(vi) UDP makes checks for errors but no reporting.
(vi) TCP gives guarantee that order of data at receiving end is same as on sending end.	(vi) UDP has no such guarantee
(vii) Header size of TCP is 20 bytes	(vii) Header size of UDP is 8 bytes
(viii) TCP is heavy weight as it needs three packets to setup connection. ~~while~~	(viii) UDP is lightweight
(IX) TCP has acknowledgement segments.	(IX) UDP has no acknowledgement
(X) TCP is used for application that require high reliability but less time critical	(X) UDP is used for application that are time sensitive but require less reliability.

ADJ13

Socket

Socket is a logical entity which provides end points to establish connection between processes or computers. Socket is a logical entity which consist IP address and Port no. to uniquely identify systems and processes over a network

Types of Sockets

Stream Sockets: It is a connection oriented socket provides reliable and sequenced delivery of information over network using TCP Protocol. Stream socket establishes a connection between client and server using TCP Protocol.

Datagram Sockets: It is a connectionless socket provides faster but unreliable and non-sequenced delivery of information over a network using UDP protocol. Datagram socket sends different packets of information over a network through all available paths.

Raw ~~Set~~ Sockets: These sockets provide users access to the underlying communication protocols, which support socket abstractions. These sockets are normally datagram oriented though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.

~~Factory methods~~

InetAddress Program

```
import java.net.*;

class InetAddressTest
{
  public static void main (String a[]) throws UnknownHostException
  {
    InetAddress Address = InetAddress.getLocalHost();
    System.out.println (Address);
    Address = InetAddress.getByName("pc24");
    System.out.println( Address)
    InetAddress sw [] = InetAddress.getAllByName(" www.
                        gpnashik.ac.in");
    for (int i=0; i< sw.length ; i++)
    {
      System.out.println (sw[i]);
    }
  }
}
```

Example of Instance Methods

```
import java.net.*;
class InetAddressInstanceTest
{
  public static void main (String a[]) throws UnknownHostException
  {
    InetAddress Address =InetAddress.getLocalHost();
    System.out.println (Address);
    byte[] adr = Address.getAddress();
    String adrstr = Address.getHostAddress();
    String hostname= Address.getHostName();
    boolean mltcst = Address.isMulticastAddress ();
    System.out.print(" Byte Address is : ");
    for (int i=0; i<adr.length ; i++)
      { system.out.println( " " + adr[i]);
      }
    System.out.println(" String Address is: " + adrstr);
    System.out.println( " Host name is: " + hostName);
System.out.println(" is multicast address " + mltcst);
  }
}
```

URL

i] The URL (Uniform Resource Locator) provides a reasonably intelligible form to uniquely identify or address information on the Internet. URLs are ~~ub~~ ubiquitos; every browser uses them to identify information on the web.

ii] Within Java's network class library, the URL class provides a simple, concise API to access information across the Internet using URLs. ~~All URLs~~

iii] All URLs share the same basic format, although some variation is allowed

eg. http://www.gpnashik.ac.in/index.html

iv] An URL specification is based on four components

1) The protocol to use, separated from rest of the locator by a colon (:)

2) The host name or IP Address of the host to use; this is delimited on the left by ~~slashes~~ double slashes (//) and on the right by a slash (/) or optionally a colon (:)

3) The port no. is an optional parameter, delimited on the left from hostname by a colon (:) and on the right by a slash (/) (It defaults to port 80, the predefined HTTP port; thus ':80' is redundant)

4) the actual file path. Most HTTP servers will append a file named index.html or index.htm to URLs that refer directly to a directory resource thus www.gpnashik.ac.in is same as www.gpnashik.ac.in/index.html

Domain Name

resource

http://www.gpnashik.ac.in:80/index.html

Protocol

Port no.

URL Class

- Constructors

i] URL (string spec)
ii] URL (string protocol, string host, int port, string file)
iii] URL (string protocol, string host, string file
iv] URL (URL contenxt, string spec)

- Methods

i] String getFile()
ii] String getHost()
iii] String getPath()
iv] int getPort()
v] String getProtocol()

URL class Program

```
import java.net.*;
class URLDemo
{
    public static void main(string a[]) throws MalformedURL
                                                Exception
    {
        URL hp = new URL("http://www.gpnashik-ac.in
                                        :80/index.html");
    System.out.println("Protocol:" + hp.getProtocol());
    System.out.println("Port: " + hp.getPort());
    System.out.println("Host:" + hp.getHost());
    System.out.println("file:" + hp.getfile());
    System.out.println("Ext :" + hp.toExternalform());
    }
}
```

—xox—

URLConnection

i] URLconnection is a general purpose class for accessing the attributes of a remote resource.
ii] Once you make a connection to a remote server, you can use URL Connection to inspect the properties of the remote object before actually transporting it locally.
iii] To get object of URLConnection class use its open connection methode on object of URL class

- URLConnection Methods
  i] int getContentLength()
  ii] String getContentType()
  iii] long getDate()
  iv] long getExpiration()
  v] String getHeaderField(int n)
  vi] String getHeaderField(String name)
  vii] InputStream getInputStream()
  viii] OutputStream getOutputStream()
  ix] URL getURL()

ServerSocket

i] The ServerSocket class is used to create servers that listen for either local or remote client programs to connect to them on published ports.

ii] ServerSockets are quite different from normal Sockets. When you create a ServerSocket, it will register itself with the system as having interest in client connections.

iii] The constructors for ServerSocket reflect the port number that you want to accept connections on and optionally, how long you want to queue for said port to be. The queue length tells the system how many client connection it can leave pending before it should simply refuse connections. The default is 50.

iv] The constructors might throw an IOException under adverse conditions.

- Constructors
  i] public ServerSocket(int port) throws IOException
  ii] public ServerSocket(int port, int backlog) throws —"—
  iii] public ServerSocket(int port, int backlog, InetAddress addr) —"—
  iv] public ServerSocket() —"—
- Methods
  i] public int getLocalPort()
  ii] public Socket accept() throws IOException
  iii] public void bind(SocketAddress host, int backlog)

Socket

i] A socket is simply an endpoint for communication between the machines

ii] The socket class can be used to create a socket.

iii] The java.net.Socket class represents the socket that both the client and the server use to communicate with each other.

iv] The client obtains a socket object by instantiating one, whereas the server obtains a socket object from the return value of the accept() method

- Constructors

i] public Socket(String host, int port) throws UnknownHostException, IOException

ii] public Socket(InetAddress host, int port) throws IOExceptiion

iii) public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException.

iv) public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException.

v) public Socket()

- Methods

i] public void connect(SocketAddress host, int timeout) throws IOException

ii] public InetAddress getInetAddress()

iii] public int getPort()

iv) public int getLocalPort()

v) public SocketAddress getRemoteSocketAddress()

vi) public InputStream getInputStream() throws IOException

vii) public OutputStream getOutputStream() throws IOException

viii) public void close() throws IOException

• URLConnection Meth...

ADJ 3

Java Security

i] The Java platform provides a no. of features designed for improving the security of Java Applications

ii] This includes enforcing runtime constraints through the use of the Java virtual Machine (JVMs), a ~~few~~ Security Manager that Sandboxes untrusted code from the rest of the operating System, and a Suite of security APIs that Java developers can utilize.

Features of Sandbox

i] Fine-grained access control
ii] Easily Configurable security policy
iii] Easily extensible access control structure
iv] Extension of security checks to all Java Programs, including application as well as applets.

• Secure Coding guidelines

• Fundamental

i] prefer to have obviously no flaws rather than no Obvious flaws.
ii] Design APIs to avoid Security concerns
iii] Avoid duplication
iv] Restrict privileges
v] Establish trust boundaries
vi] Minimize the no. of permission checks
vii] Encapsulate
viii] Document Security related information.

• Denial of service

i] Beware of activities that may use ~~disappro~~ disproportionate resources
ii] Release resources in all cases
iii] Resource limit checks should not suffer from integer overflow

• confidential Information

i] Purge sensitive information from exceptions
ii] Do not log highly sensitive information
iii] Consider purging highly sensitive from memory after use.

Socket primitives

Socket primitives defines different stages or methods through which socket performs different operations during communication between hosts

SOCKET : Create a new communication end point
BIND : Attach a local address to a socket.
LISTEN : Announce willingness to accept connections; give queue size
ACCEPT: Block the caller until a connection attempt arrives
CONNECT : Actively attempt to establish a connection
SEND : Send some data over the connection
RECEIVE : Receive some data from the connection
CLOSE : Release the connection

———— x o x ————

Port

It is a 16 bit unique number used to uniquely identify processes over a network. It is also called as a numbered socket. A port number is the logical address of each application or process that uses a network or the Internet to communicate. A port no. uniquely identities a network-based application on a computer. Each application/program is allocated a 16-bit integer port no. This number is assigned automatically by the OS, manually by the user or is set as a default for some popular applications. ~~Max~~ Maximum port no.s : 65536 i.e. 0 to 65535

Types of ports

i] well known/reserved ports : Some port numbers are reserved for standard protocols and applications by IANA are called as well known reserved ports. These numbers are pre-assigned to standard protocols and no one can use these for general purposes. Total reserved ports are 1024 i.e 0 to 1023

ii] Registered ports : Some port numbers can be registered by software companies for their network applications or specific protocols are called as Registered ports. Range of ports available for registration is 1024 to 49151

iii] Dynamic or private ports : Port no. 49152 to 65535 are available for general purposes or research purposes are called as dynamic or private ports. These port ~~no~~ numbers can be used by anybody for their private use.

——— XOX ———

InetAddress

i] InetAddress encapsulates both numerical IP address and the domain name for that address

ii] InetAddress can handle both IPv4 and IPv6 address

iii] InetAddress has no visible constructor

iv] ~~InetAd~~ To create an InetAddress object, you have to use Factory methods.

Factory methods

Factory methods are the static methods of any class that returns the object of same class where these methods are defined. Factory methods of InetAddress class returns the instance of InetAddress class because it does not provides any visible constructor.

Factory methods of InetAddress class

i] static InetAddress getLocalHost() throws UnknowHostException
ii] static InetAddress getByName(String hostName) —||—
iii] static InetAddress getAllByName(String hostName) —||—
iv] static InetAddress getByAddress(byte[] addr) —||—

Instance Methods

Instance methods are the non-static method of any class which are defined in that class and accessed only by object of that class. Instance method means member methods of object of a class. Instance methods of InetAddress returns IP Address information of any host stored by its objects.

Instance methods of InetAddress class

i] boolean[] ~~get~~ equals(Object other)
ii] byte[] getAddress()
iii] String getHostAddress()
iv] String getHostName()
v] boolean isMulticastAddress()
vi] String toString()

- Accessibility and Extensibility
  i] Limit the accessibility of classes, interfaces, methods and fields
  ii] Limit the accessibility of packages
  iii] Isolate unrelated code
  iv] Limit exposuse of classloader instances
  v] Limit the extensibility of classes and methods
  vi] Understand how a superclass can affect subclass behaviour.
- Input validation
  i] validate inputs
  ii] validate output from untrusted objects as input
  iii] Define wrappers around native methods
- Socket Programming

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket (node) listens on a particular port at an IP, while other socket reaches out to other in order to form a connection. The server forms the listener socket while the client reaches out to the server. socket and server socket classes are used for connection oriented socket programming

]

Port no-s

Oracle : 1521
SQLServer : 1433

- Steps for Database connectivity
  i] Loading the driver
  ii] Create the connection
  iii] Create Statement
  iv] Execute the Query
  v] close the connection

—— x o x ——

- Two Tier Database Design

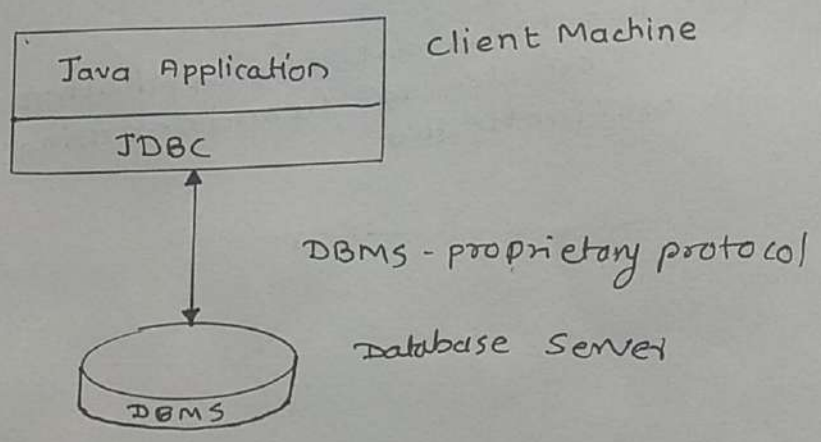


Fig. Two Tier Database Design.

i] The two tier is based on client server architecture. The tier architecture is like client server application.
ii] The direct communication takes place between client and server. There is no intermediate between client and server.
iii] Because of tight coupling a 2tired application will run faster
iv] In two tier architecture database design user interface programs and application programs for database access run on client machine.
v] The interface program called JDBC provides an API that allows application programs to call DBMS through JDBC drivers.

The two tier architecture is divided in two parts

i) Client Application (Client Tier)
ii) Database (Data Tier)

On client application side the code is written for saving the data in SQL server database. client sends the request to server and it process the request and send back with data. the main problem of two tier architecture is the server cannot respond multiple request same time, as a result it cause a data integrity issue

- Advantages
  i) Easy to maintain and modification is bit easy
  ii) communication is faster
- Disadvantages
  i) In two tier architecture application performance will be degrade upon increasing the users.
  ii) Cost-ineffective

———————xOx———————

Three Tier Database Design

ADT134

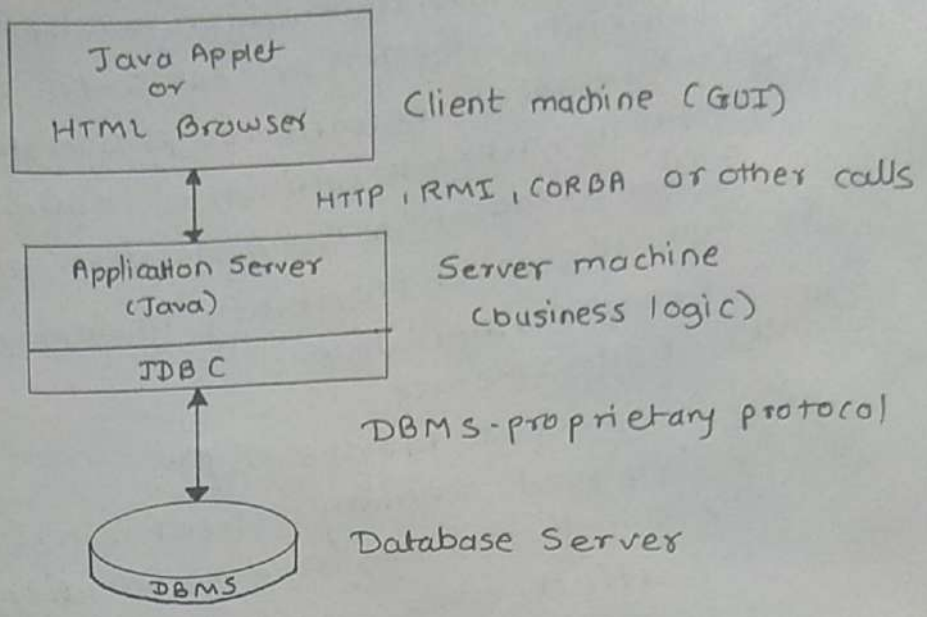


Fig. Three-tier Database Design

Three-Tier architecture typically comprises a presentation layer, a business or data access tier and a data layer.

i) client Layer

It is also called as Presentation layer which contains UI part of our application. This layer is used for the design purpose where data is presented to the user or input is taken from the user.

For eg. designing registration form which contains textbox, label, button etc.

ii) Business layer

In this layer all business logic written likes validation of data, calculations, data insertion etc. This acts as an interface between client layer and Data access Layer. This layer is also called the intermediary layer helps to make communication faster between client and data Layer.

iii) Data Layer

In this layer actual database comes into the picture. Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data.

• Advantages

i] High performance, lightweight persistent objects
ii] Scalability - Each tier can scale horizontaly
iii] Performance - Because the presentation tier can cache requests, network utilization is minimized and the load is reduced on the Application and Data tiers.
iv] High degree of flexibility in deployment platform and configuration.
v] Better Re-use
vi] Improve Data Integrity.
vii] Improved Security - client is not direct access to database
viii] Easy to maintain and modification is bit easy, won't affect other modules
IX] In three architecture application performance is good.

• Disadvantages

i] Increase Complexity / Effort

JDBC

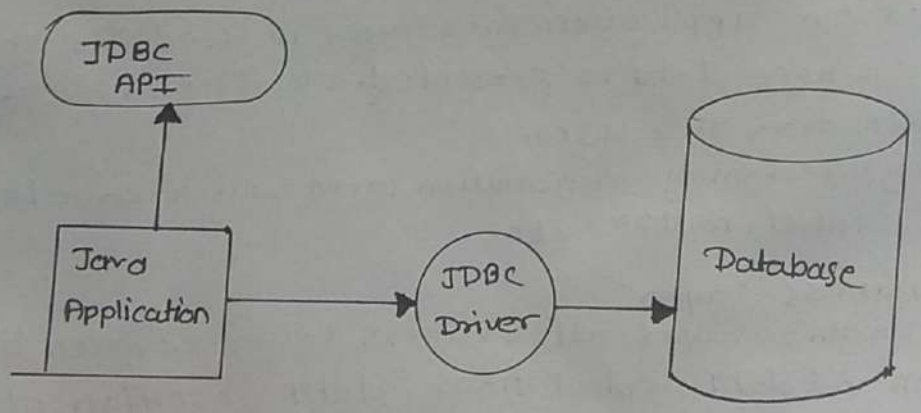


Fig. JDBC

i] Java Database Connectivity (JDBC) is an application program interface (API) specification for connecting programs written in Java to the data in popular databases.
ii] The application program interface lets you encode access request statements in Structured Query Language (SQL) that are then passed to the program that manages the database.
iii] It returns the results through a similar interface.
iv] JDBC is a Java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database.

- JDBC Features
- Get a connection : A java application can be connected to a database either using DriverManager or DataSource object.
- connection pooling: It allows the java application to reuse database connection that has been created already instead of creating a new connection every time.
- Rowsets : The rowsets object contains the tabular data. It makes the possible to pass the rows data to the network. Therefore they are widely used in distributed application.
- New dataType supports: This is the ability of JDBC to manipulate large object such as BLOB and CLOB without bringing them to the java programmer from the database server.
- Batch Updating: This feature provides the ability to send multiple updates to the database to be executed as batch rather than sending each update separately
- Result set enhancement
  - Scrollable Resultset : It provides the ability to move the cursor backward and forward to a specific position
  - updateable Resultset: It allows the modification of data in database table using result set.
- Savepoints : JDBC contains a savepoint interface which contains a new method to set a savepoint, to release a savepoint to rollback a transaction to desired savepoints.

JDBC Architecture

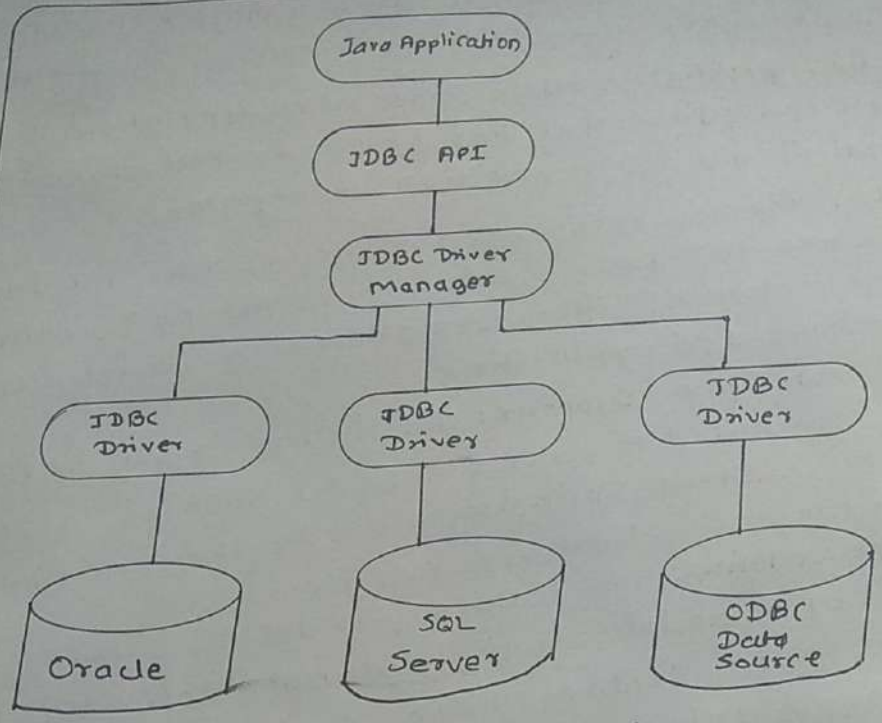


Fig. JDBC Architecture

The JDBC API Supports both two-tier and three-tier processing models for database access but in general JDBC Architecture consists of two layers

- JDBC API: It provides application to JDBC Manager Connection
- JDBC Driver API: This supports the JDBC to Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases. The JDBC driver manger ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases

JDBC Components

i] DriverManager
ii] Driver
iii] Connection
iv] Statement
v] ResultSet
vi] SQLException

connect with the database. It uses jdbc drivers to

• DriverManager Class

The DriverManager class acts as an interface between users and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. The DriverManager class maintains a list of Driver classes that have registered themselves by calling the method DriverManager.registerDriver()

• Methods

i) public static void registerDriver (Driver driver)
ii) public static void deregisterDriver (Driver driver)
iii) public static Connection getConnection (String url)
iv) public static Connection getConnection (String url, String userName, String password)

Driver Interface

Driver is the interface that every driver class must implement. The Java SQL Framework allows for multiple database drivers. Each driver should supply a class that implements the Driver interface. The DriverManager will try to load as many drivers as it can find and then for any given connection request, it will ask each driver in turn to try to connect to target URL.

Connection interface

A connection is the session between java application and database. The connection interface is a factory of statement, PreparedStatement and DatabaseMetadata. i.e. object of connection can be used to get the object of statement and DatabaseMetaData. The Connection interface provide many methods for transaction management like commit(), rollback() etc. By default, connection commits the changes after executing the queries.

Statement interface

The statement interface provides methods to execute queries with the database. The statement interface is a factory of ResultSet i.e. it provides method to get object of ResultSet.

• Prepared Statement

The PreparedStatement interface is a subinterface of statement. It is used to execute parameterized query.

eg. String sql = "insert int emp values(?,?,?)";

As you can see, we are passing parameter (?) for the values. Its value will be set by calling the setter methods of Prepared Statement.

Improves performance: The performance of the application will be faster if use PreparedStatement interface because query is compiled only once.

• Callable Statement

To call the stored procedures and functions, CallableStatement interface is used. We can have business logic on the database by the use of stored procedures and functions that will make performance better because these are precompiled.

Suppose you need to get the age of employee based on the date of birth, you may create a function that receives date as the input and returns age of the employee as the output

Diffrence between stored procedure and functions

Stored Procedure	Function
(i) It is used to perform business logic	(i) It is used to perform calculation
(ii) It must not have return type	(ii) It must have the return type.
(iii) may return 0 or more values	(iii) It return ~~only~~ only one value
(iv) we can call function from procedure	(iv) Procedure cannot be called from function
(v) ~~Proce~~ Procedure supports input and output parameters	(v) function support only input parameter
(vi) Exception handling using try/catch block can be used in stored procedure	(vi) Exception handling using try/catch can't be used in user defined functions

- Resultset

The object of ResultSet maintains a cursor pointing to a particular row of data. Initially, cursor points to before the first row. By default, Resultset object can be moved forwardonly and it is not updateable. But we can make this object to move forward and backward direction by passing either TYPE_SCROLL_INSENSITIVE or TYPE_SCROLL_SENSETIVE in CreateStatement (int, int) Method as well as we can make this object updatable by:

```
Statement stmt = con.CreateStatement (Resultset.TYPE_
SCROLL_INSENSITIVE, Resultset. CONCUR_UPDATABLE);
```

- ResultsetMetaData

i] public int getColumnCount () throws SQLException
ii] public String getColumnName(int index) —//—
iii] public String getColumnTypeName (int index) —//—
iv] public String getTableName (int index) —//—

—xo—

JDBC Drivers.

A JDBC driver is a software component enabling a Java application to interact with a database. To connect with individual databases, JDBC requires drivers for each database. The JDBC driver gives out the connection to the database and implements the protocol for transfering the query and result between client and database.

There are 4 types of JDBC drivers

1) JDBC-ODBC bridge driver
2) Native-API driver (partially java driver)
3) Network Protocol driver (fully java driver)
4) Thin driver (fully java driver.

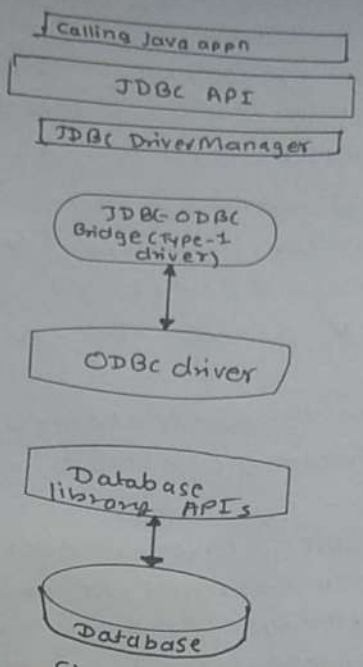


Fig. JDBC-ODBC bridge driver

i] The Type 1 driver translates JDBC calls into ODBC calls and sends them to ODBC driver.

ii] ODBC is a generic API. the JDBC-ODBC bridge driver is recommended only for experimental use or when no other alternative is available.

iii] In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine.

iv] Using ODBC requires configuring on your system a Data Source Name (DSN) that represents the target database. when Java first came out, this was a useful driver because most databases only supported ODBC access but now this type of driver is recommended only for experimental use or when no other alternative is available

- Advantage

The JDBC-ODBC Bridge allows access to almost any database, since the database's ODBC drivers are already available

- Disdvantges

i] since the bridge driver is not written fully in Java, Type 1 drivers are not portable

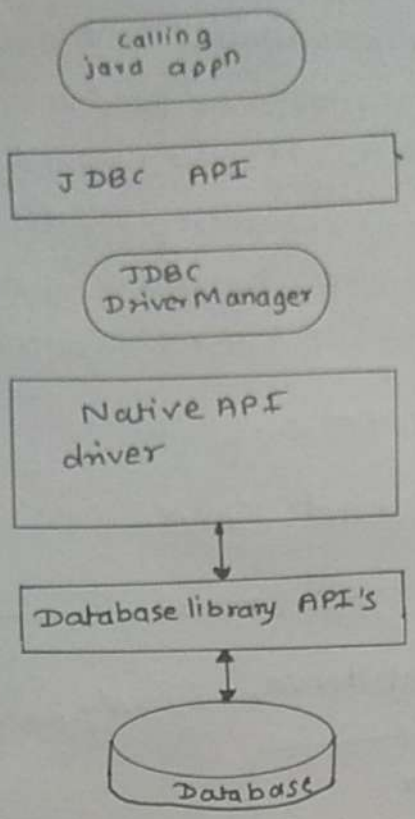
ii] A performance issue is seen as a JDBC call goes through the bridge to ODBC, then to database, in reverse process also. They are slowest of all driver types

iii] The client system requires the ODBC installation to use driver

iv] Not good for the web.

2. Native - API driver (partially Java Driver)

ADJ14



i] the distinctive characteristic of type 2 ~~d~~ ~~jbd~~ jdbc drivers is that Type 2 drivers convert JDBC calls into database specific calls i.e. this driver is specific to a particular database

ii] eg. Oracle will have oracle native API. In a Type 2 driver, JDBC API calls are converted into native C/C++ API calls which are unique to the database. These drivers typically provided by the database vendors and used in the same manner as the JDBC-ODBC Bridge, the vendor specific drives must be installed on each client machine.

iii] If we change the Database we have to change the native API as it is specific to a database and they are mostly obsolete now but you may realize some speed increase with a Type 2 driver because it eliminates ODBC overhead.

• Advantage

The distinctive characteristic of type 2 jdbc drivers is ~~are~~ that they ~~are~~ typically offer better performance than the JDBC-ODBC bridge as the layers of communication (tiers) are less than that of Type 1 and also it uses Native API which is Database specific.

• Disadvantages

i] Native API must be installed in the client system and hence type 2 drivers cannot be used for the Internet

ii] Like Type1 drivers, its not written in Java language which forms a portability issue

iii] If we change the Database we have to change the native API as it is specific to a database

iv] Mostly obsolate now

v] Usually not thread safe.

---

3. Network Protocol driver (fully java driver)

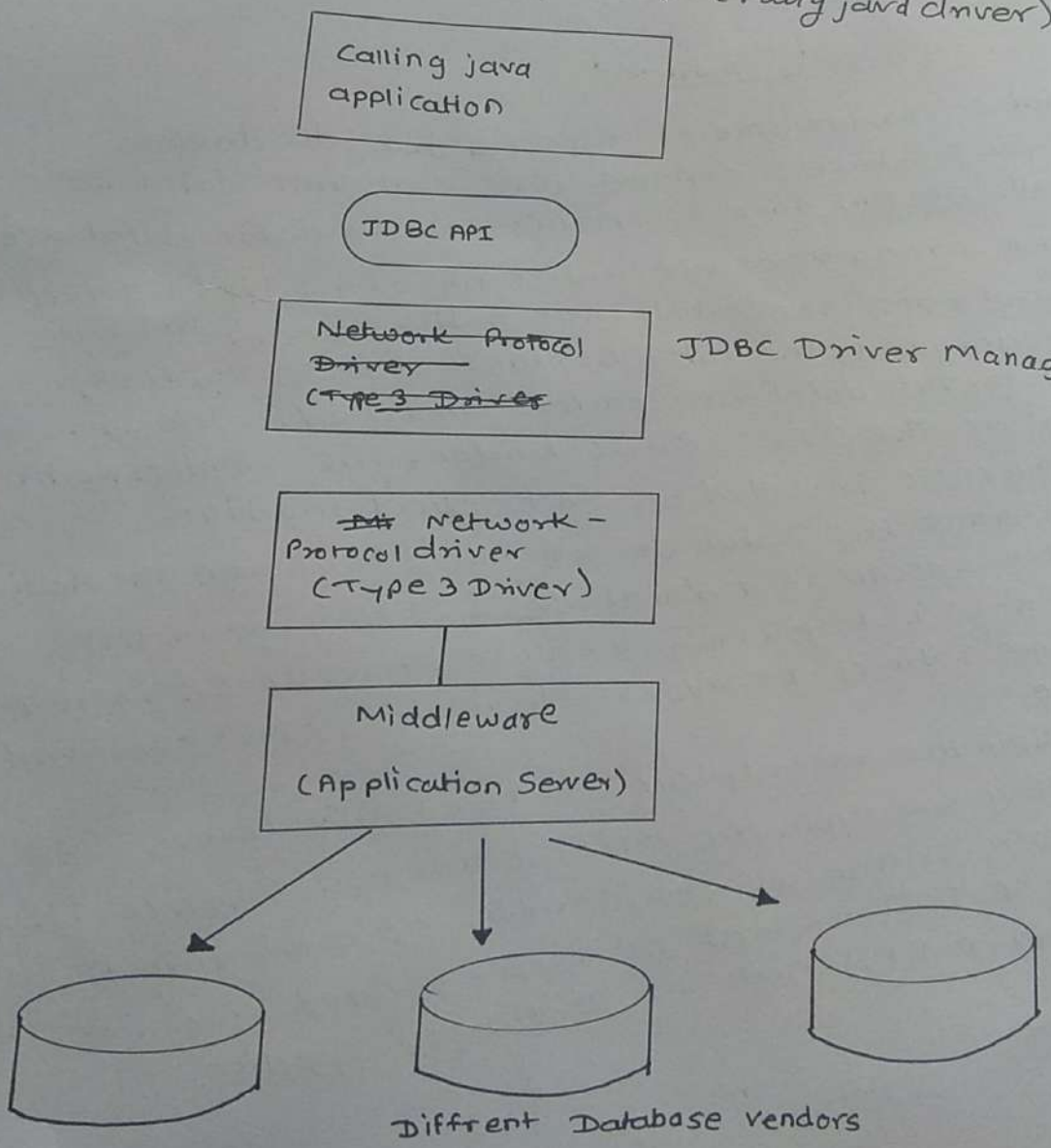


fig. Network Protocol driver

3. Network Protocol driver

i] Type 3 database requests are passed through the network to the middle tier server. ~~the~~ The middle tier then translates the request to the database.

ii] If the middle tier server can in turn use Type1, Type2 or Type4 drivers. In a Type 3 driver, a three tier approach is used to access databases. The JDBC clients use standard network sockets to communicate with a middle application server.

iii] The socket information is then translated by the middleware application server into the call format required by the DBMS, and forwarded to the database server.

iv] this kind of driver is extremely flexible, since it requires no code installed on the client machine and a single driver can actually provide access to multiple databases

- Advantages

i] This driver is server-based, so there is no need for any vendor database library to be present on client machine.

ii] This driver is fully written in Java and hence portable. It is suitable for the web.

iii] there are many opportunities to optimize portability, performance and scalability

iii] The net protocol can be designed to make the client JDBC driver very small and fast to load.

iv] The Type 3 driver typically provides support for features such as caching (connections, query results, and so on), load balancing and advanced system administration such as logging and auditing.

v] This driver is very flexible allows access to multiple databases using one driver

vi] They are most efficient amongst all driver types

- Disadvantage

It requires another server application to install and maintain. Traversing the record set may take longer, since the data comes through the backend server.

...in driver (fully Java driver)

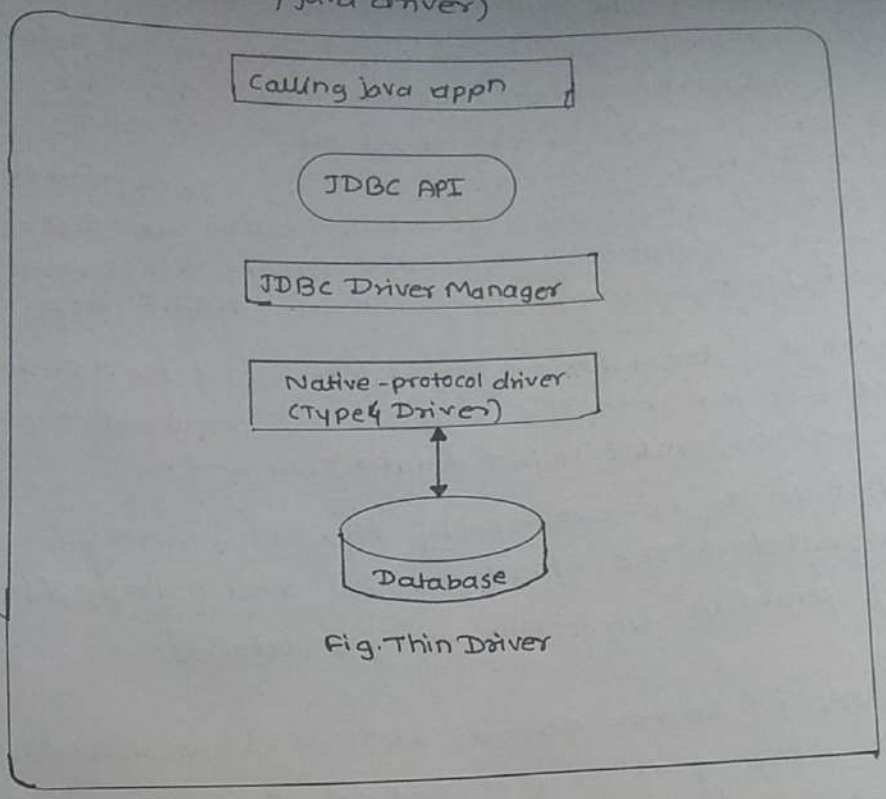


Fig. Thin Driver

i] The Type4 uses Java networking libraries to communicate directly with database server

ii] In a Type4 driver, a pure Java-based driver that communicates directly with vendor's database through socket connection.

iii] This is the highest performance driver available for the database and is usually provided by the vendor itself.

iv] This kind of driver is extremely flexible; you don't need to install special software on the client or server

v] Further, these drivers can be downloaded dynamically.

• Advantage

i] The major benefit of using type 4 jdbc drivers are that they are completely written in Java to achieve platform independence and eliminate administration issues. It is most suitable for web.

ii] Number of translation layers is very less i.e. Type4 JDBC drivers don't have to translate database requests to ODBC or a native connectivity interface or to pass on to the another server. performance is typically quite good.

iii] You don't need to install special software on the client or server. Further these ~~softwares~~ drivers can be downloaded dynamically

• Disadvantage

With typ4 drivers, the user needs a different driver for each database.

...driver

- Steps to connect to the database in java

i] Register the driver class

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class

Syntax: public static void forName(String className) throws ClassNotFoundException

eg. Class.forName("oracle.jdbc.driver.OracleDriver");

ii] Create the connection object

The getConnection() method of DriverManager class is used to establish connection with the database

syntax: • public static Connection getConnection(String url) throws SQLException

• public static Connection getConnection(String url, String name, String password);

eg. Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "password");

iii] Create the Statement object

The createStatement() method of connection interface is used to create statement. The object of statement is responsible to execute queries with the database

Syntax: public statement createStatement() throws SQLException

eg. Statement stmt = con.createStatement();

iv] Execute the Query

The executeQuery() method of statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table

Syntax: public ResultSet executQuery(String sql) throws SQLException

eg. ResultSet rs = stmt.executeQuery("select * from emp");
while(rs.next()) {
System.out.println(rs.getInt(1) + " " + rs.getString(2));
}

v] Close the Connection object

By closing Connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax: public void close() throws SQLException

eg. con.close();

ADJ14

Purpose of JDBC API

i] To access tables and its data from relational database

ii] To send queries and update statement to database

iii] obtain and modify the results ~~and~~ to and from a JDBC application

iv] find the metadata of table

v] Performing different operations on a database like creating table, querying data, updating data, inserting data from a Java application.

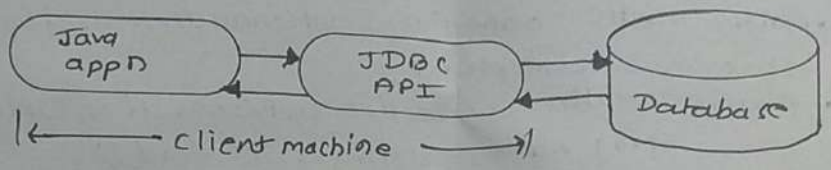


fig-Two tier Architecture of JDBC

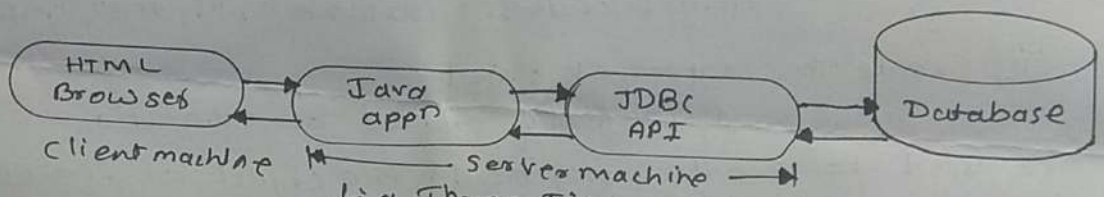


fig-Three-Tier architecture of JDBC

Browser : A browser fetches a page from a web server by a request. A request is a standard request containing a page address. An address may like this: http://www.example.com/a.html All web pages contain instructions for display. The browser displays the page by reading these instructions. The most common instructions are called HTML tags. HTML tags look like this

 This is a paragraph

Web Application : A web application is an application accessible from the web used to generate dynamic responses. A web application is composed of web components like servlet, JSP, Filter etc. and other components such as HTML. The web components typically execute in web server and respond to HTTP server.

HTTP Features

i] HTTP is connectionless
ii] HTTP is media independent
iii] HTTP is stateless

Request Methods

i] GET
ii] HEAD
iii] POST
iv] PUT
v] DELETE
vi] CONNECT
vii] OPTIONS
viii] TRACE

Diffrence between GET and POST

GET	POST
(i) In case of Get request, only limited amount of data can be sent because data is sent in header	(i) In case of post request, large amount of data can be sent because data is sent in body
(ii) Get request is not secured because data is exposed in URL bar	(ii) Post request is secured because data is not ~~secured~~ exposed in URL bar.
(iii) Get request can be bookmarked	(iii) Post request cannot be bookmarked.
(iv) Get request is idempotent. It means ~~ea~~ Second request will be ignored until response of first request is delivered.	(iv) Post request is non-idempotent
(v) Get request is more efficient and used more than Post. It is default method.	(v) Post request is less efficient and used less than GET. It is not default method.

Status codes

i] 1xx : Informational request was recieved and process is continuing
ii] 2xx : Success action was succesfully received
iii] 3xx : Redirection further action must be taken in order to complete request
iv] 4xx : Client Error request contain incorrect syntax can't be fulfilled
v] 5xx : Server Error It means server failed to fulfill an apparently valid request.

MIME : Multipurpose internet mail extension

Content Type

i] text/html
ii] text/plain
iii] application/msword
iv] application/vnd.ms-excel
v] application/jar
vi] application/pdf
vii] application octet-stream
viii) application/x-zip
ix] images/jpeg
x] video/quicktime

Web Server

A web server is a program that, using the client/server model and the world wide web's Hypertext Transfer Protocol (HTTP), serves the files that form web pages to web users. Two leading web server are Apache and IIS.

Application Server

An application server is a program that handles all application operations between users and an organization's backend business application or database.

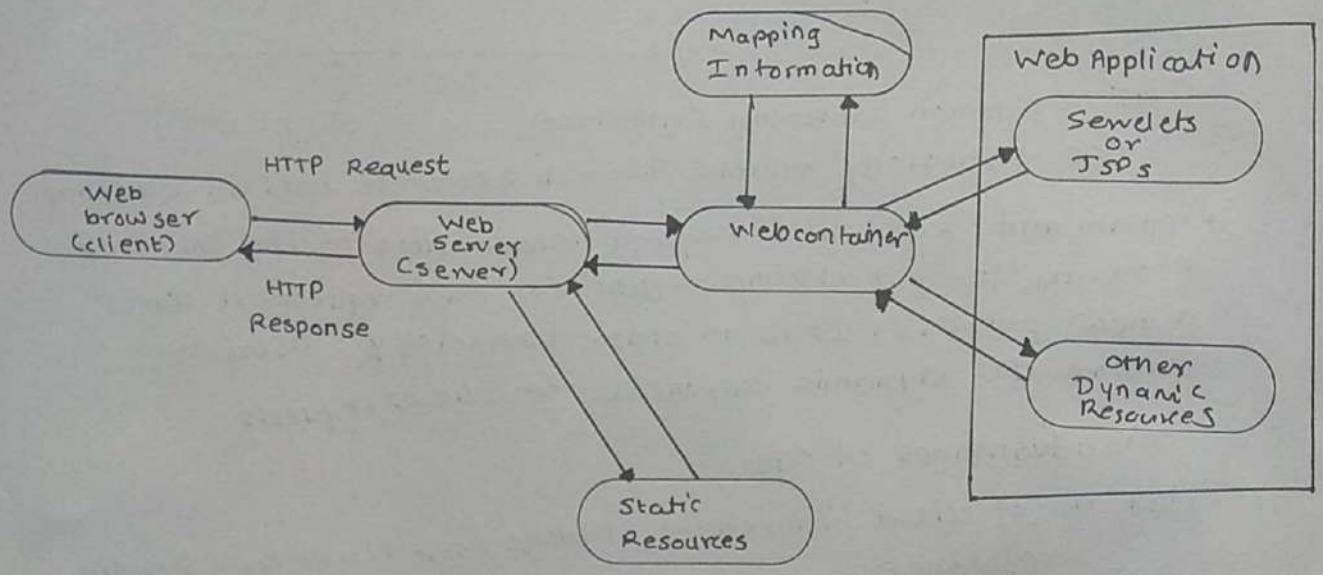


Fig. Working of Web Server

Web container

A web container is a J2EE complient implementation which provides an environment for the servlets and JSP to run.

Servlet

i) Servet is a technology used to create web application
ii) Servlet is an API that provides many interfaces and classes including documentations
iii) servlet is an interface that must be implemented for creating any servlet. ~~prog~~
iv) Servlet is a class that extends the capabilities of servers and responds to incoming request. It can respond to any type of request.
v) Servlet is a web component that is deployed on server to ~~cred~~ create dynamic web page

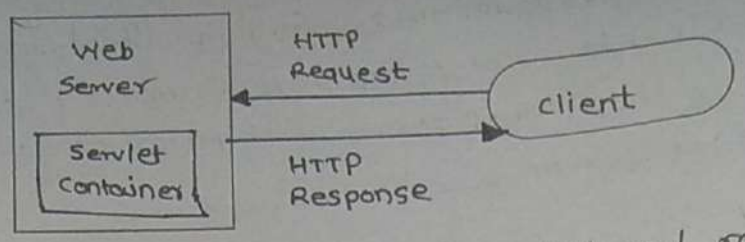


Fig. Servlet request and response

Servlet container performs
i] Lifecycle Management
ii] Multithreaded Support
iii] Object Pooling
iv] Security.

CGI (Common Gateway Interface)
CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request it starts a new process. It is an older technology used to generate dynamic responses for client requests.

Disadvantages of CGI
(i) If no. of client increases, it takes more time for sending response
(ii) For each request, it starts a process and web server is limited to start processes.
(iii) It uses platform dependent language eg. c, c++, perl

Advantages of servlet
(i) Better Performance: because it creates a thread for each request not process.
(ii) Portability: Servlets are platform-independent because they are written in Java.
(iii) Robust: Servlets are managed by JVM so no need to worry about memory leak, garbage collection etc.
(iv) Secure: because it uses java language
(v) Servlets execute within the address space of a web server. It is not necessary to create separate process to handle each client request.
(vi) Java Security manager on the server

enforces a set of restrictions to protect the resources on a server machine. So servlets are trusted

vii) The full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases, or other other software via the Socket and RMI mechanisms that you have seen already.

Servlet Life Cycle

A servlet life cycle can be defined as the entire process from its creation till the destruction.

i] The servlet is initialized by calling init() method

ii] The servlet calls service() method to process a client's request.

iii] The servlet is terminated by calling the destroy() method.

iv] Finally servlet is garbage collected by garbage collector of JVM.

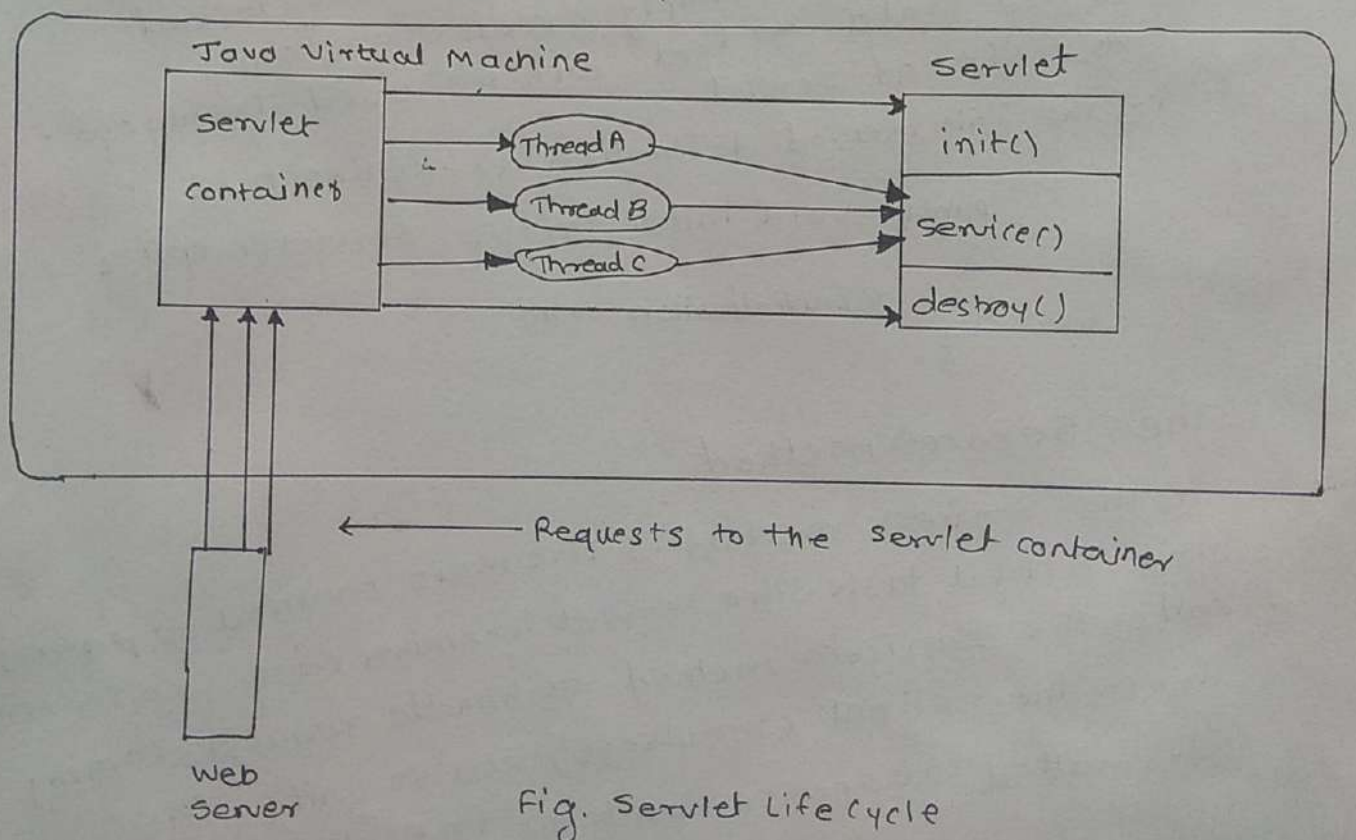


Fig. Servlet Life Cycle

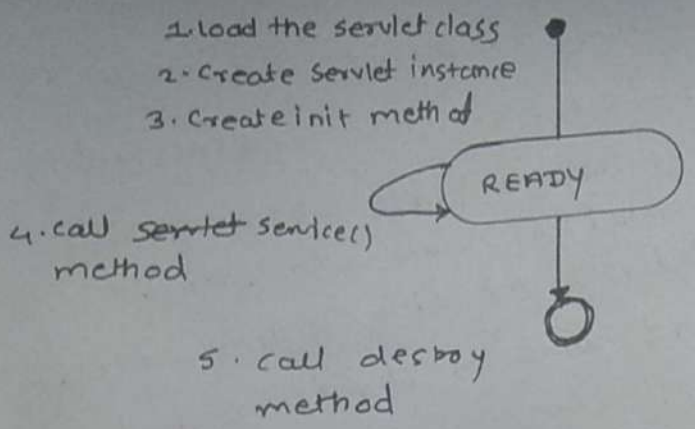


fig. Servlet Lifecycle

The init() method

i] The init method is designed to be called only once. It is called when the servlet is first created, and not again for each user request.

ii] So, it is used for onetime initializations, just as with init method of applets.

iii] The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet to be loaded when the user request resulting in new thread. The init() method simply creates or loads some data that will be used througout the life of servlet

iv] The init method detination looks like this

```
public void init() throws ServletException
{
    //Initialization code
}
```

The service() method

i] The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client (browsers) and to write the formatted response back to the client.

ii] Each time the server receives a request for a servlet, the server spawns a new thread and calls service.

iii] The service() method checks the HTTP request type (GET, POST, PUT, DELETE) and calls doGet, doPost, doPut, doDelete etc. methods as appropriate.

iv] Here is signature of that method

```
public void service (ServletRequest request,
    ServletResponse response) throws servletException,
                                     IOException
{
    //generate response to client
}
```

v] The service() method is called by container and service method invokes doGET, doPost, doPut, doDEleete etc. methods as appropriate. The doGet() and doPost() aremost freaquently used methods within each service request.

The destroy() method

i] The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk and perform other such cleanup activities. After the destroy() method is called, the servlet object is marked for garbage collection.

ii] The destroy() method detination look like this

```
public void destroy()
{
    //Finalization code
}
```

The javax.servlet package

Interface: i] Servlet
ii] ServletResponse
iii] ServletRequest
iv] RequestDispatcher
v] ServletConfig

classes: i] GenericServlet
ii] ServletInputStream
iii] ServletOutputStream
iv] ServletRequestWrapper
v] ServletResponseWrapper

——— X o X ———

Servlet Interface

i] servlet interface provides common behaviour to all the servlets. interface needs to be implemented for creating any servlet (either directly or indirectly)

ii] It provides 3 life cycle methods that are used to initialize the servlet, to service the requests and to destroy servlet and 2 non-life cycle methods

• methods

i] public void init(ServletConfig config)
ii] public void service(ServletRequest request, ServletResponse response)
iii] public void destroy()
iv] public ServletConfig getServletConfig()
v] public string getServletInfo()

- ServletRequest

An object of ServletRequest is used to provide the client request information to servlet such as content type, content length, parameter names and values, header informations, attributes etc.

- ServletResponse

The ServletResponse interface contains various methods that enable a servlet to respond to the client requests. A servlet can send the response either as character or binary data. The PrintWriter stream can be used to send character data as servlet response. and ServletOutputStream stream to send binary data as servlet response.

GenericServlet class

GenericServlet class implements Servlet, ServletConfig and Serializable interfaces. It provides the implementation of all methods of these interfaces except the service method. GenericServlet class can handle any type of request so it is protocol independent. You may create a generic servlet by inheriting the GenericServlet class and providing implementation of the service() method.

The javax.servlet.http package

Interfaces

i] HttpServletRequest
ii] HttpServletResponse
iii] HttpSession
iv] HttpSessionListener
v] HttpSessionAttributeListener

Classes

i] HttpServlet
ii] Cookie
iii] HttpServletRequestWrapper
iv] HttpServletResponseWrapper
v] HttpSessionEvent

- HttpSession interface

i] the Httpsession interface enables a servlet to read and write the state information that is associated with an http session

ii] Httpsession object is used to store entire session with a specific client.

iii] we can store, retrieve and remove attribute from Httpsession object.

iv] Any servlet can have access to HttpSession object through ~~out~~ the getsession() method of the HttpServletRequest object.

- Httpservlet class

i] the HttpServlet class extends the GenericServlet class and implements Serializable interface. Http servlet is also an abstract class. ~~This class gives~~

ii] This class gives implementation of various service() methods of servlet interface.

iii] To create a servlet, we should create a class that extends HttpServlet abstract class.

iv] The servlet class that we will create, must not override service() method. our servlet class will override only the doGet() and/or doPost() methods.

~~v) The service~~

v] The service() method of HttpServlet class listens to the Http methods (GET, POST etc) from request stream and invokes doGet() or doPost() methods based on Http method type.

• Cookies

i] Cookies are small piece of information that are sent in response from web server to the client.

ii] Cookies are the simplest technique used for storing client state.

iii] Cookies are stored on client's computer. They have a lifespan and are destroyed by client browser at the end of that lifespan. It is information for future that is stored by the server on the client side of a client/server communication.

iv] Cookies are usually small text files, given ID tags that are stored on your computer's browser directory or program data subfolders.

v] Cookies are created using Cookie class present in servlet API.

vi] Cookies are added to response object using the addCookie() method. This method sends cookie information over the HTTP response stream.

vii] getCookies() method is used to access cookies that are added to response object

Advantages of cookies

i] simplest technique of maintaining the state

ii] Cookies are maintained at client side

Disadvantages of cookies

i] It will not work if cookie is disabled from the browser

ii] Only textual information can be set in cookie object.

Constructors

i] Cookie()

ii] Cookie(String name, String value)

- Steps to create a servlet
  i] Create a directory structure
  ii] Create a Servlet
  iii] Compile the servlet
  iv] Create a deployment descriptor
  v] Start the server and deploy the project

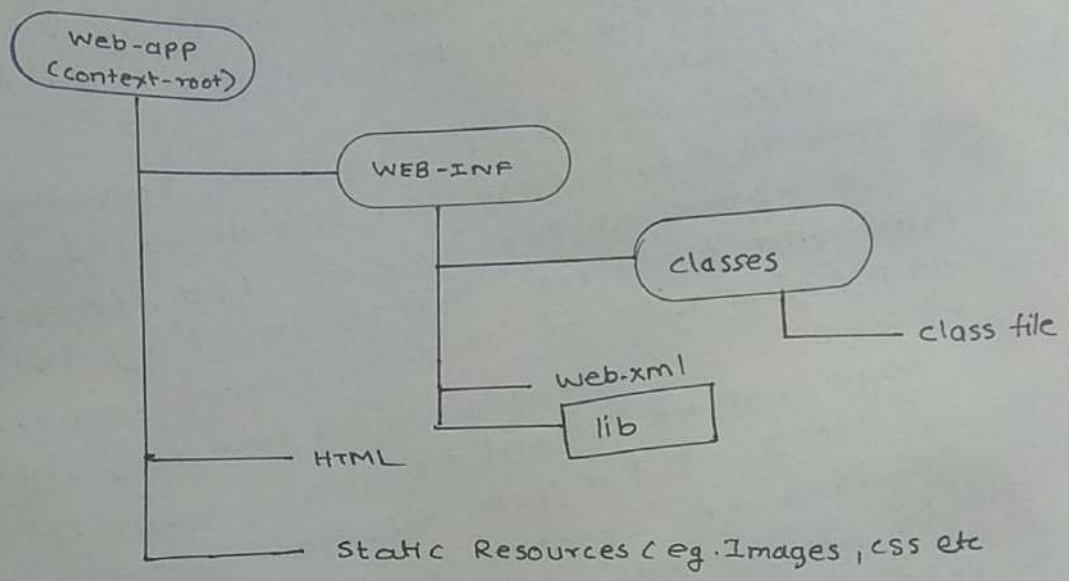


Fig. Directory Structure

- There are three ways to create the Servlet
  i] By implementing the Servlet interface
  ii] By inheriting the Generic servlet class
  iii] By inheriting the HttpServlet class.

The HttpServlet class is widely used to create servlet because it provides methods to handle http requests such as doGet(), doPost, doHead() etc.

- To compile Servlet
  i] Download servlet-api.jar
  ii] paste servlet-api.jar file inside Java\jdk\jre\lib\ext folder
  iii] use javac

ADJIS

web.xml file

i] <web-app> : ~~the~~ represents the whole application
ii] <servlet> : is sub element of <web-app> and represents the servlet.
iii] <servlet-name>: is sub element of <servlet> ~~represents~~ represents name of servlet
iv] <servlet-class>: is sub-element of <servlet> represents the class of servlet
v] <servlet-mapping>: is sub element of <web-app>. It is used to ~~ma~~ map the servlet.
vi] <url-pattern>: is sub element of <servlet-mapping>. This pattern is used at client side to invoke servlet.

- How Servlet works

The server checks if the servlet is requested for the first time.

If yes, web container does following tasks
i] Loads the servlet class
ii] instantiates the servlet class
iii] calls the init method passing the ServletConfig Object

else

- calls the service method passing request and response objects

The web container calls the destroymethod when it needs to remove the servlet such as at time of stopping server or undeploying the project.

- How web container handles the servlet request?

i] maps the request with the servlet in web.xml file
ii] Creates request and response objects for this request.
iii] Calls the service method on the thread
iv] The public service method internally calls the protected service method
v] the protected service method calls the doGet method depending on the type of request.
vi] The doGet method generates the response and it passed to the client
vii] After sending the response, the web container deletes the request and response objects. The thread is contained in the thread pool or deleted depends on the server implementation.

Diffrence between forward() and SendRedirect() method

forward() method	sendRedirect() method
i] The forward() method works at server side	i] The sendRedirect() method works at client side
ii] It sends the same request and response objects to another response	ii] It always sends a new request.
iii] It can work within server only	iii] It can be used within and outside server.
iv] eg. request.getRequestDispatcher("servlet2").forward(request, response);	iv] eg. response.sendRedirect("servlet2");

- Session

i] The session of activity that a user with a unique IP address spends on a web site during a specified period of time. The no. of user sessions on site is used in measuring the amount of trafting a web site gets

ii] session: an abstract concept to represent a series of HTTP requests and responses between a specific web browser and server. HTTP doesn't support the notion of a session but Java does.

- Session tracking

i] Session tracking is the capability of a server to maintain the current state of a single client's sequential requests.

ii] The HTTP protocol used by web servers is stateless. This means that every transaction is autonomous.

iii] This type of stateless transaction is not a problem unless you need to know the sequence of actions a client has performed while at your site.

iv] Session management is a mechanism used by the web container to store session information for a particular user.

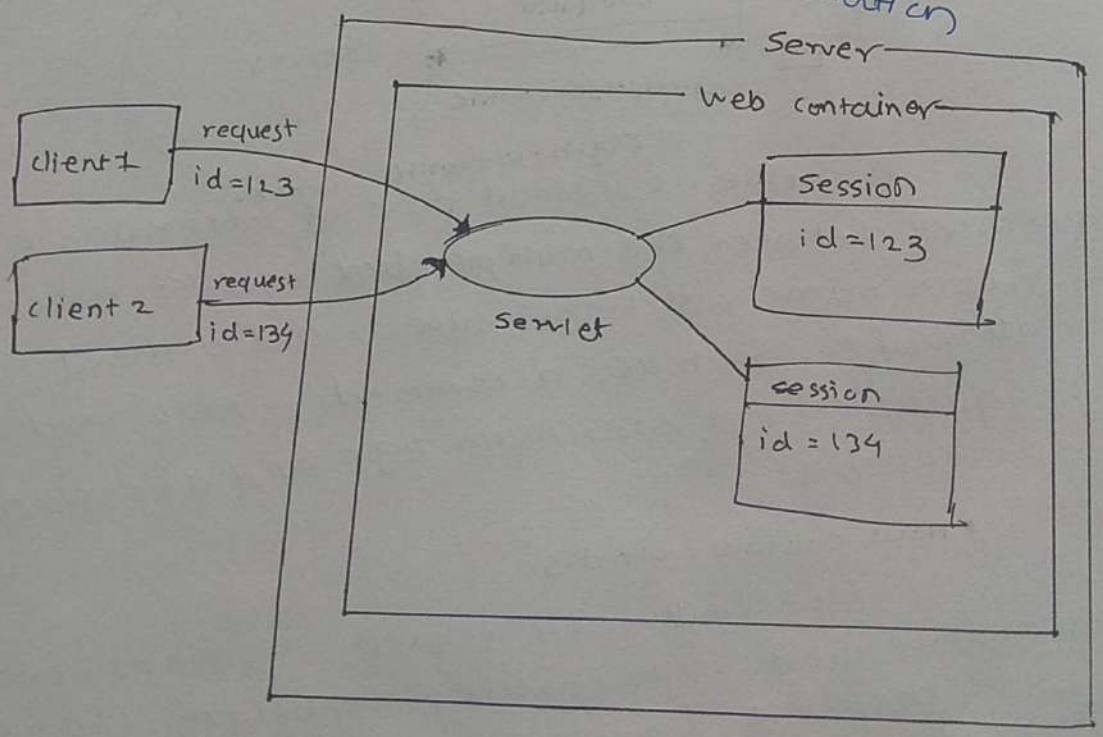


Fig. How Session works

- Session Tracking Techniques

i] Cookies: You can use HTTP cookies to store information. cookies will be stored at browser's side

ii] URL rewriting: With this method, the information is carried through url as request parameters. In general added parameter will be sessionid, userid

iii] Http session: Using Http session, we can store information at server side. Http session provides methods to handle session related information.

iv] Hidden form fields: By using hidden form fields we can insert information in the webpages and these information will be sent to the server. These fields are not visible directly to the user. but can be viewed using view source option from browsers. The hidden fields are given below

```
<input type="hidden" name="varname" value="val"/>
```

- Cookies

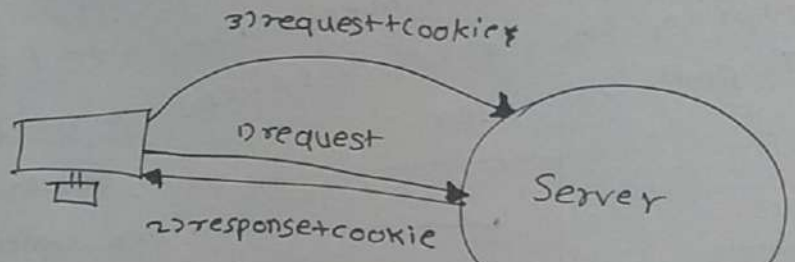


fig. How cookie works

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

- How cookie works

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 it request is sent by user, cookie is added with request by default. Thus we recognize the user as old user.

- Importance of JSP

i) Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages. itself instead of having a Separate CGI file.

ii] JSP are always compiled before its processed by the Server unlike CGI/Perl which requires the server to load an interpretor and the target script each time the page is requested.

iii] Java Server Pages are built on top of the Java Servlets API, so like servlets; JSP also has access to all the powerful ~~Pages~~ Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.

iv] JSP Pages can be used in combination with servlets that handle the business logic, the model & Supported by Java Servlet template engines

- Advantages of JSP

i] Extension to servlet
ii] Easy to maintain
iii] Fast Development (No need to recompile and redeploy)
iv] Less Code than Servlet.

- JSP

i] Compilation
ii] Initialization
iii] Execution
iv] Cleanup

- JSP Compilation

i] parsing the JSP
ii] Turning JSP into a Servlet
iii] Compiling the servlet.

~~in~~ Simple JSP Page

```
<html>
    <body>
        <%= out.print (2*5); %>
    </body>
</html>
```

Output: 10

- The javax.servlet.jsp package

I] Interfaces

i] JspPage
ii] HttpJspPage

II] classes

i] JspWriter
ii] PageContext
iii] JspFactory
iv] JspEngineInfo
v] JspException
vi] JspError.

According to the JSP specification, all the generated servlet classes must implement the JspPage interface. It extends Servlet interface.

- Scripting elements of jsp

i] scriptlet tag
ii] expression tag
iii] declaration tag

JSP implicit Objects

There are 9 jsp implicit objects

Object	class
out	JspWriter
request	HttpServletRequest
response	HttpServlet Response
config	ServletConfig
application	ServletContext
session	HttpSession
PageContext	PageContext
page	Object
exception	Throwable

- JSP Directives

  i] Page directive
  ii] include directive
  iii] taglib directive

i] Page directive

Syntax : <%@ Page attribute = "value" %>

eg : <%@ page import = "java.util.Date" %>

ii] include directive

Syntax: <%@ include file = "resourceName" %>

eg: <%@ include file = "header.html" %>

iii] taglib directive

Syntax: <%@ ~~tag~~ taglib uri = "uriofthe taglibrary"
prefix="prefixoftaglibrary" %>

eg. : <%@ taglib uri = "http://www.gpnashik.ac.in/tags"
prefix = "mytag" %>

- Action tags

There are many JSP action tags or elements Each jsp action tag is used to perform some specific tasks. The action tags are used to control flow between pages and to use java bean.

eg. jsp: ~~forward~~ ⇒ forwards request and response to anothe resource

jsp: include ⇒ include another resource.

diffrence between JSP include directive and JSP include action tag

JSP include directive	JSP include action tag
i] includes resource at translation time	i] include resource at request time
ii] better for static pages	ii] better for dynamic pages
iii] includes the original content in the generated servlet.	iii] calls the include method

ADJIS

ADJIS

- JavaBean

A Java Bean is a java class that show follow following conventions

i] It should have a no-argument constructor
ii] It should be serializable
iii] It should provide methods to set and get the values of the properties, known as getter and setter methods

Simple example of java Bean class

```
~~Packag~~
package mypack;
public class Employee implements java.io.Serializable
{
    private int id;
    private string name;
    public Employee() {}
    public void setId(int id) {this.id =id;}
    public void getId() { return id;}
    public void setName (String name) {this.name =name;}
    public String getName() { return name;}
}
```

- JSP Expression Language

The Expression language (EL) simplifies the accessibility of data stored in the Java Bean component and other objects like request, session, application etc. There are many implicit objects, operators and reserved words in EL.
It is the newly added feature in JSP technology version 2.0

Syntax: ~~$$~~ ${expression}

- JSTL

JSP Standard Tag Library (JSTL) is a standard library of readymade tags. The JSTL contains several tags that can remove scriptlet code from a JSP page by providing some ready-to use, already implemented common functionalities

Advantages of JSTL
i] Fast Developement
ii] Code Reusability
iii] No need to use scriptlet tag

LAD315

- JSTL is divided into 5 groups

  i] JSTL Core
  ii] JSTL Formatting
  iii] JSTL sql
  iv] JSTL XML
  v] JSTL functions

- JSP custom tags

  Custom tags are nothing but user-defined tags. Custom tags are an excellent way to abstract the complexity of business logic from presentation of web pages in a way that is easy for the web author to use and control. It also allows for reusability as custom tags can be used again and again.

  Advantages of custom tags

  i] Eliminates the need of scriptlet tag
  ii] Separation of business logic from JSP
  iii] Reusability.

RMI

i] The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java.

ii] The RMI allows an object to invoke methods on an object running in another JVM.

iii] The RMI provides remote communication between the applications using two objects stub and skeleton.

iv] RMI is a way that a programmer, using th java programming language and developement environment, can write object-oriented programming in which objects on diffrent computers can interact in a distributed network.

v] RMI is the Java version of what is generally known as a remote procedure call (RPC), but with ability to pass one or more objects along with the request.

vi] The object can include information that will change the service that is performed in the remote computer. Remote Method Invocation facilitates function calls between Java virtual machines (JVMs)

ADJ IS

ADJ IS

RMI is nothing but a communication between two JVMs placed on different systems.

The RMI System consists of three layers

1] The stub / skeleton layer or proxy layer

2] The Remote Reference Layer

3] The Transport Layer

Stub is a client side proxy

Skeleton is a server side proxy

Marshaling: Marshaling is nothing but converting data into a special format suitable to pass through the distributed environment without losing object persistance.

Un-marshalling: Un-marshalling is the process of extracting marshaled data into its original format.

Stub performs the marshalling, whereas skeleton performs the un-marshaling

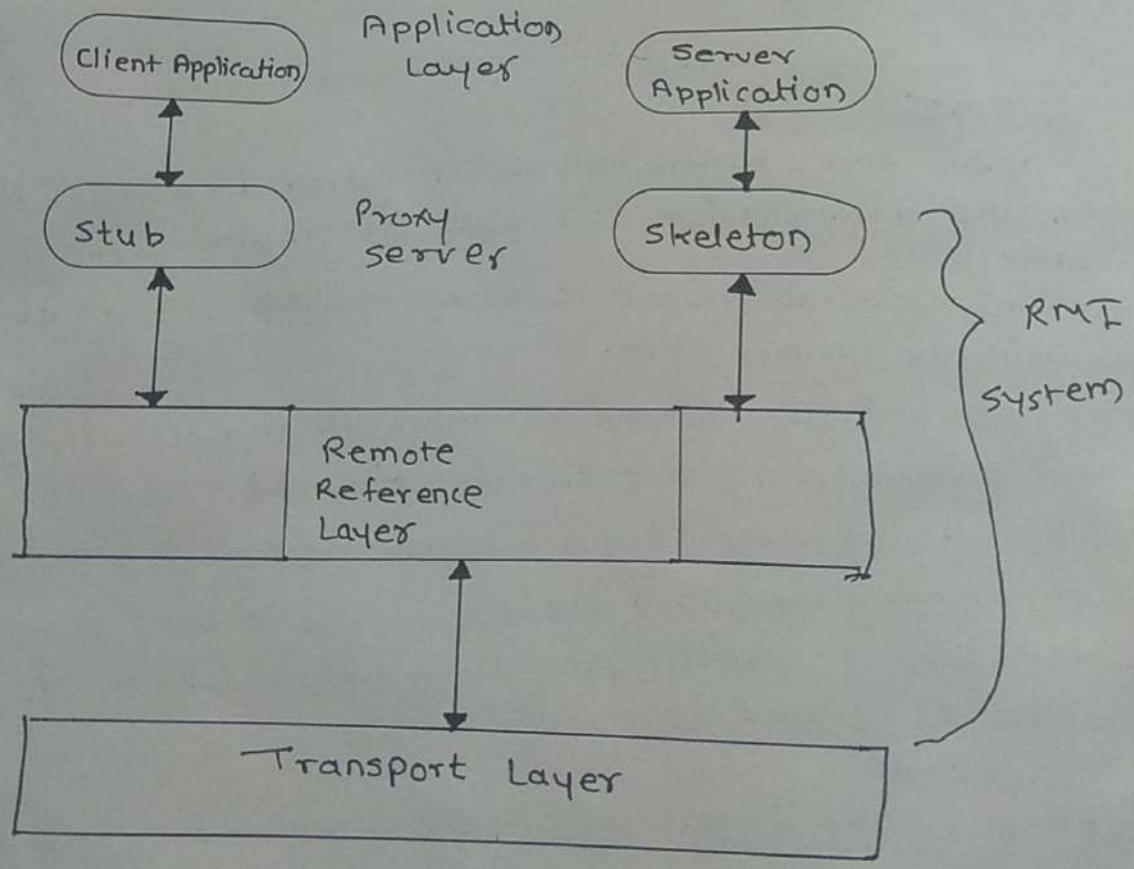


Fig. Architecture of RMI

- Advantages of RMI

i] Handles threads and sockets for underlying communication.
ii] Server-side implementation can be changed without the knowledge of the client side.
iii] To extend an RMI solution, you can extend or add new classes, just like in a non-distributed application.

- Disadvantages of RMI

i] Overhead of marshaling and un-marshaling
ii] Overhead of object serialization
iii] Cannot use the code out of scope of java.

Default port for RMI registry is 1099

- Steps to create RMI application

i] Create the remote interface
ii] provide the implementation of remote interfaces
iii] Create server application and register service with registry.
iv] Create client application

- Steps to execute RMI application

i] Compile all the java files
ii] Create stub and skeleton object by rmic tool
iii] Start rmi registry in one command prompt
iv] Start server in another command prompt
v] Start client appn in another command prompt.

EJB

i] EJB is an acronym for enterprise java bean.
ii] It is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.
iii] Enterprise Java Beans (EJB) is a managed, Server-side component architecture for modular construction of enterprise applications.
iv] EJB is a Server side model that encapsulates the business logic of an application.

ADJ IS
ADJ IS
ADJ IS

✓] The EJB Specification intends to provide a standard way to implement the back-end 'business' code typically found in enterprise applications.

- Features of EJB

i] Transaction processing
ii] Integration with persistance services offered by the Java Persistence API (JPA)
iii] Concurrency Control
iv] Event driven programming using Java Message Service and Java EE Connector Architecture.
v] Asynchronous method invocation
vi] Job Scheduling
vii] Naming and directory services (JNDI)
viii] Interprocess Communication using RMI-IIOP and web services
~~viii~~ IX] Security (JCE and JAAS)
X] Deployment of Software Components in application Server
XI] EJB Components are serverside components written entirely in the Java programming language.
XII] EJB architecture is inherently transactional, distributed, portable, multi-tier, scalable and secure.
XIII] EJB components are fully portable across any EJB server and any OS.
XIV] EJB architecture is wire protocol neutral -- any protocol can be utilized like IIOP, HTTP, JRMP, DCOM etc.

Types of EJB

i] Session Bean
ii] Entity Bean
iii] Message Driven Bean

- Benefits of EJB

i] simplified developement of large scale enterprise level application

ii] Application Server/EJB Container provides most of the system level services like transaction handling, logging, load balancing, persistence mechanism, exception handling and so on. Developer has to focus only on business logic of the application.

iii] EJB container manages lifecycle of ejb instances thus developer needs not to worry about when to create/delete ejb objects.

- Limitations of EJB

i] Requires Application server

ii] Requires only java client. For other language client, you need to go for web service.

iii] Complex to understand and develop EJB applications.

Both RMI and EJB provide services to access an object running in another JVM. from another JVM

Diffrence between RMI and EJB

RMI	EJB
i] In RMI, middleware services such as security, transaction management, object pooling etc. need to be done by java programmer.	i] In EJB middleware services are provided by EJB container automatically.
ii] RMI is not a server-side component. It is not required to be deployed on server	ii] EJB is a server-side component; it is required to be deployed on the server.
iii] RMI is built on top of socket programming	iii] EJB technology is built on top of RMI