OBJECT ORIENTED CONCEPTS AND PROGRAMMING – II (ADVANCE JAVA)

DR. BABASAHEB AMBEDKAR OPEN UNIVERSITY
AHMEDABAD

Editorial Panel

Authors : **Deepal Shah**
Assistant Professor,
J. G. College of Computer Application,
Ahmedabad

&

Dr. Vinod Desai
Assistant Professor,
Department of Computer Science,
Gujarat Vidyapith, Ahmedabad

Editor : **Dr. Himanshu N. Patel**
Assistant Professor,
School of Computer Science,
Dr. Babasaheb Ambedkar Open University,
Ahmedabad

Language Editor : **Dr Mrinalini P. Thaker,**
Bhavan's Seth R.A. College of Arts & Commerce,
Ahmedabad

ISBN 978-93-91071-09-7

Edition : 2021

Acknowledgment

Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.

The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.

ROLE OF SELF–INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author (s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self–instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self–instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual–skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face–to–face method of teaching. This is

particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self–instructional materials, be they print or otherwise. These materials are designed to achieve certain pre–determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included many tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

OBJECT ORIENTED CONCEPTS AND PROGRAMMING – II (ADVANCE JAVA)

Contents

Unit 7 XML

Introduction, Application of XML, Well formed and valid XML documents, XML Namespace, XML Parser, Document type definition (DTD), XML schema

---

BLOCK 3 : RMI & JAVABEANS

Unit 8 REMOTE METHOD INVOCATION

Introduction, RMI Architecture, How RMI works, Steps to create an RMI Application, Steps for deploying RMI Application (on same system), Troubleshooting, Advanced RMI Concepts

Unit 9 JAVABEANS – 1

Introduction, Java Beans Concepts, The Beans Development Kit, Writing a Simple Bean

Unit 10 JAVABEANS – 2

Introduction, Properties, Simple Properties, Bound Properties, Indexed Properties, Constrained Properties

Unit 11 JAVA NAMING AND DIRECTORY INTERFACE API

Introduction, Naming and Directory Service, Enter JNDI, JNDI Overview, Understanding the Concepts behind JNDI Programming, Programming with JNDI, Exploring Javax.naming package

---

BLOCK 4 : SERVLETS & JSP PROGRAMMING

Unit 12 SERVLETS

Introduction, Servlet Types & Life Cycle, Servlet API, Threading Issues, Session Tracking, Writing and Running Servlet application in Apache Tomcat 7, Request Dispatcher

Unit 13 INTRODUCTION TO STRUTS

Introduction, Life Cycle of Struts Request and it's Component Class, Struts Action Classes, Struts Model Components, The Struts View Components, Configuring the web.xml file for Struts, Writing and Executing Struts Application

Unit 14 JSP (JAVA SERVER PAGES)

Introduction, JSP Life Cycle, JSP Architecture, JSP Basic Building Blocks, JSP Implicit Objects, Standard Actions, JSP Tag Libraries

Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-304

OBJECT ORIENTED CONCEPTS AND PROGRAMMING – II (ADVANCE JAVA)

BLOCK 1 : JAVA REVIEW & SWING COMPONENTS

JAVA REVIEW & SWING COMPONENTS

Block Introduction :

Java is a programming language created by James Gosling from Sun Microsystems in 1991. The first publicly available version of Java was released in 1995 as Java 1.0. An applet is a small component written in Java that can be included or "plugged" into an HTML page or other markup language page. Swing is a framework for developing the graphical user interface (GUI) for Java applications and applets.

In this block, we will detail about the basic of Java Platform Features and Java Programs and Components. The block will focus on the study and concept of Java 2 Platform Editions and Java Platform Environment. The students will give an idea on Swing Component and Containment Hierarchy.

In this block, the student will made to learn and understand about the basic of Layout Management techniques. The concept related to Event Handling and Setting Path and Classpath will also be explained to the students. The student will be demonstrated practically about Writing a Swing Program technique.

Block Objectives :

After learning this block, you will be able to understand:

- About Swing Component and Containment Hierarchy
- Basic of Java Platform Features
- Features of Java 2 Platform Editions
- Concept of Event Handling
- Detailed about History of Java Platform
- Basic of Writing a Swing Program
- Idea of Java Programs and Components

Block Structure :

Unit
01

INTRODUCTION, HISTORY AND JAVADOC

UNIT STRUCTURE

1.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Java Programs and Components
- History of Java Platform
- Java 2 Platform Editions

1.1 Introduction :

Java is a programming language created by James Gosling from Sun Microsystems in 1991. The first publicly available version of Java was released in 1995 as Java 1.0. This is commonly used programming language that has variety of uses and applications which makes them to used in desktop applications, Mobile Applications, Enterprise applications etc.

Java is :

- Class Based and Object Oriented Programming Language
- Computing platform
- Fast, Secure and Reliable
- Free
- General Purpose
- Concurrent

1.2 Java Programs and Components :

Programming is writing of instructions sets which will guide the computer how to do certain work. Doing work relates to reading list of names from a file which could be alphabetical and writing it back again to the file. As seen, it could be much more complex as it involves displaying of graphical

user interface which could mainly meant for games and other game's logic. Since it is analysed that Java is relatively a current object-oriented programming language which has nowadays achieved more popularity and is easy to apply. It is a general-purpose language that can be used for many types of programming tasks.

The programs that were written and designed up till now carries a sequential flow of control. It means that the statements initially were executed line by line from top to bottom in a particular order. It is found that when any statement happens to skip or executed more than once, then it needs a control. In this unit we will see that such can be done using control structures. The control structure carries three groups :

- Decision making statements
- Repetition statements
- Branching statements

Decision making is type of control statements that will decide whether or not a given statement or group of statements be worked out or not. Repetition statements are such where the statements are carried out more than ones. Branching statements on the other hand transfers the control to another line in a given code.

Java component :

The architectures of Java carry special features of Java language and component-based model for software development. The basic components of the architecture are:

Java applets :

Components built using the Java programming language that can be dynamically downloaded across the network and executed directly from within an HTML container. Java applet components are typically compiled to bytecode, which is interpreted by a Java run-time environment.

JavaScript scripts :

Components built using the JavaScript scripting language that can be dynamically downloaded across the network and interpreted from within an HTML container.

❑ **Check Your Progress – 1 :**

1. Programming language involves :

   a. computer b. codes c. software d. all

1.3 History of Java Platform :

Java was initially framed by Sun Microsystems in the beginning of 1990s.

The language was created with the idea to solve problem related to connectivity of several household machines together.

As it was designed for special connectivity purpose only, so the particular project was failed as no one wants to use it.

Initially the name of Java is OAK.

James Gosling worked initially on Java and was known as father of Java.

The name OAK was renamed as Java in the year 1994.

During the mid of year 1995 in month of May, Java was publicly released.

Java was targeted for Internet development.

The release of applets was initially supported by big companies such as Netscape Communications.

Some of the Java Versions are shown in table 1.1.

Java Version	Release Date	Year
$JDK_{1.0}$	January 21	1996
$JDK_{1.1}$	February 19	1997
$J_2SE_{1.2}$	December 8	1998
$J_2SE_{1.3}$	May 8	2000
$J_2SE_{1.4}$	February 6	2002
$J_2SE_{5.0}$	September 30	2004
Java SE_6	December 11	2006
Java SE_7	July 28	2011

Table 1.1 Versions of Java

❑ **Check Your Progress – 2 :**

1. OAK was renamed as Java in year :

   a. 1995 b. 1996 c. 1997 d. 1994

1.4 Javadoc Comments :

Javadoc is a tool that appears along with JDK and it is used for generating Java code documentation in HTML format from Java source code that carries required documentation in predefined format.

Javadoc comments are multi-line comments ("/** ... */") which are kept before class, field or method declarations which begin with slash and two stars and carries special tags to show characteristics involving method parameters or return values. The HTML files generated by Javadoc will describe each field and method of a class, using the Javadoc comments in the source code itself. It is made up of description part which is followed by block tags such as tags like @param, @return, and @see.

```
**
*   Returns an Image object that can then be painted on the screen.
*   The url argument must specify an absolute {@link URL). The name
*   argument is a specifier that is relative to the uri argument.


*<p>
*   This method always returns immediately, whether or not the
*   image exists. When this applet attempts to draw the image on
*   the screen, the data will be loaded. The graphics primitives
*   that draw the image will incrementally paint on the screen.
```

```
*     @param url an absolute URL giving the base location of the image
*     @param name the location of the image, relative to the url argument
*     @return the image at the specified URL
*     @see Image *
public Image getImage(URL url, Sting name) {
      try {
         return getImage(newURL(url. name));
      } catch (MalformedURLException e} {
         return null;
      }
}
```

We see that normally, Javadoc comments be placed before any class, field, or method declaration so as to describe its intent or characteristics. Consider an example :

```
/**
*     Represents a student enrolled in the school.
*     A student can be enrolled in many courses.
*
public class Student {

**
*     The first and last name of this student.
*/
private String name;
**
*     Creates a new Student with the given name.
*     The name should indude both first and

*     last name.
*/
public Student(String name) {
      this, name = name;
      }
}
```

❑ **Check Your Progress – 3 :**

1. Javadoc comments are placed before:

   a. any class b. any field

   c. any method declaration d. all of above

1.5 Let Us Sum Up :

In this unit we have learnt that Java is a programming language created by James Gosling from Sun Microsystems in 1991. The first publicly available version of Java was released in 1995 as Java 1.0.

Java is an object-oriented language where data in application and methods can be manipulated instead of procedures. It is seen that in an object-oriented system, class is a collection of data and methods which operates on data.

It is noted that programming is writing of instructions sets which will guide the computer how to do certain work which can be reading list of names from a file which could be alphabetical and writing it back again to the file.

The Java Platform Micro Edition which is also known as Java ME, is a platform which is designed for particular embedded systems.

It is seen that Java 2 Platform Enterprise Edition also called as J2EE shows a component based mechanism so as to construct, develop, assemble and deploy enterprise applications.

An application runs in a platform environment, defined by the fundamental operating system, Java virtual machine, class libraries and different arrangement data complete when the application is launched.

1.6 Answer for Check Your Progress :

- **Check Your Progress 1 :**

  1. (d)

- **Check Your Progress 2 :**

  1. (d)

- **Check Your Progress 3 :**

  1. (d)

1.7 Glossary :

1. **Version :** It is a sort of description or an account that differs from one point of view to another.
2. **Bytecode :** It is a computer object code which runs through a program known as virtual machine instead by actual computer machine.

1.8 Assignment :

Write a program which will print "PASS" if int variable "mark" is more than or equal to 60 else it will print "FAIL".

1.9 Activities :

Discuss the output obtained from this program.

```
int a=new int[l0]:
int i,s=0;
for(i=0;i<10;i++)
{
a[i]=i;
s=s+a[i]+i;
```

```
}
System.out.println (s);
```

1.10 Case Study :

Discuss about the Java program shown ?

```
public class CheckOddEven { //saved as "CheckOddEven.java
  public static void main (String[] args) {
    int number = 49;        //set the value of number here!
    System.out.println("The number is "+number};
    if(  ){
      System.out println(.....);
    } else {
      System.out println(.....);
      }
    }
  }
```

1.11 Further Readings :

1. A Primordial Interface by Wm. Paul Rogers
2. Java Diamonds Forever by Tony Sintes
3. Java Routines by John D. Mitchell
4. Object-Oriented Design and Programming by Tony Sintes

Unit
02

JAVA PLATFORM, SETTING AND CLASSPATH

UNIT STRUCTURE

2.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Writing a Swing Program
- Swing Component and Containment Hierarchy
- Layout Management

2.1 Introduction :

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on

Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

Advance java is a part of Java programming language. It is an advanced technology or advance version of Java specially designed to develop web-based, network-centric or enterprise applications. It includes the concepts like Servlet, JSP, JDBC, RMI, Socket programming, etc. It is a specialization in specific domain.

Most of the applications developed using advance Java uses two-tier architecture i.e. Client and Server. All the applications that runs on Server can be considered as advance Java applications.

2.2 Java Platform Features :

Java is an object-oriented language where data in application and methods can be manipulated instead of procedures. It is seen that in an object-oriented system, class is a collection of data and methods which operates on data. Taken together, data and methods shows state and behavior of an object. Classes are arranged in hierarchy, so that subclass can inherit behavior from its superclass. Java comes with an extensive set of classes, arranged in packages that you can use in your programs.

We see that platform is the base where software/hardware along with program runs. It is found that Java has software called JVM that runs on software platform which is an Operating system. As Java runs on any of platform but it is required to have software JVM. So programs can be developed in any platform and can run in any platform.

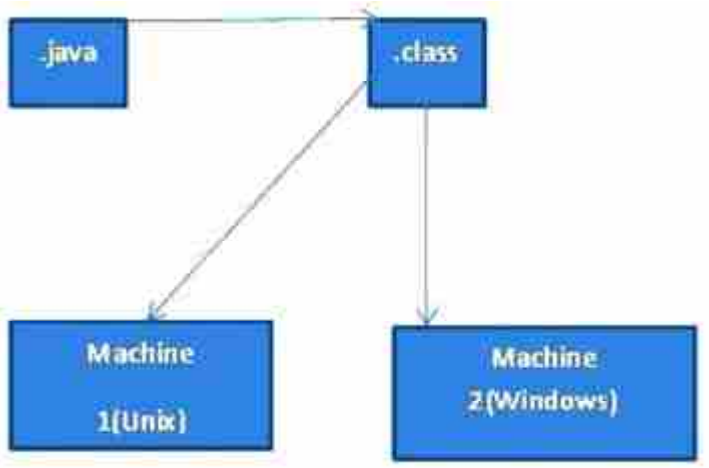


Figure 1 Java Platform

We see that the binary form of programs running on Java platform is not a local machine code but serves as intermediate bytecode. The JVM will carry out verification on such bytecode before running it to save program from performing unsafe operations which includes branching to incorrect locations having data rather than instructions.

It also allows JVM to implement runtime constraints such as array bounds checking that serves as an important less likely to suffer from memory safety flaws like buffer overflow with comparison to programs written in certain languages that do not have such memory safety guarantees.

We see that the platform will not allow programs to do certain potentially unsafe operations like pointer arithmetic or unchecked type casts. Further, also it does not allow manual control over memory allocation and deallocation; users are required to rely on the automatic garbage collection provided by the platform. This also contributes to type safety and memory safety.

❑ **Check Your Progress – 1 :**

1. The name of Java software is :

   a. C++ b. JVM c. Oracle d. none of above

2.3 Java 2 Platform Editions :

Java is formally called as Java 2 Platform which carries three editions :

- Java 2 Standard Edition (J2SE)
- Java 2 Enterprise Edition (J2EE)
- Java 2 Micro Edition (J2ME)

Figure 2 Java Editions

All these three editions will focus on different kinds of applications which runs on different devices. It is found that :

- Desktop based applications were developed with the help of J2SE that carries necessary user interface classes.
- Server based applications was formed with the help of J2EE that stress more on component based programming as well as deployment.
- Handheld along with embedded devices are formed with the help of J2ME.

In the year 1995, the JDK 1.0 exists which was upgraded to JDK 1.1 and to Java 2 in the year 1999.

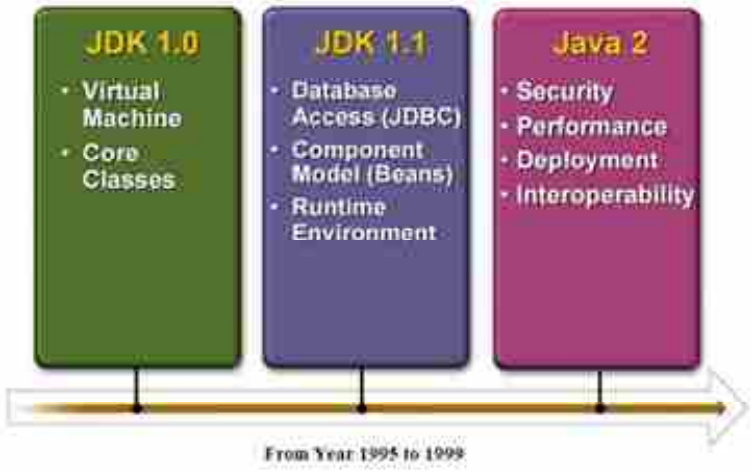


Figure 3 JDK

It is seen that there exists single Java platform with multiple profiles such as :

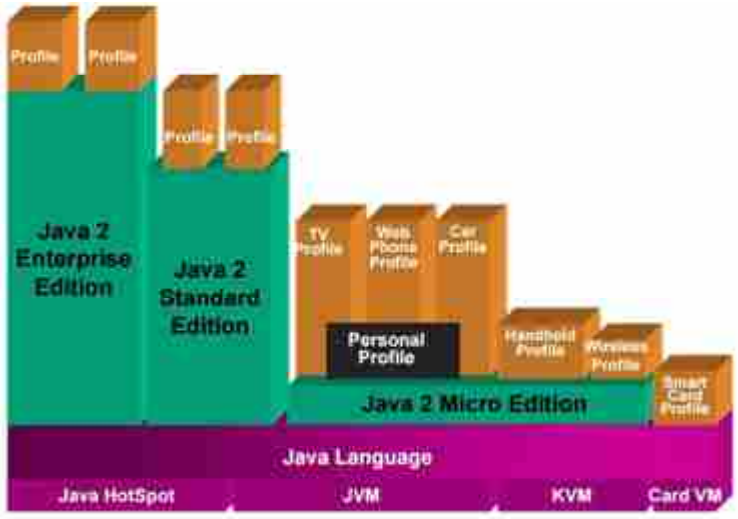


Figure 4 Java Platforms

J2ME

The Java Platform Micro Edition which is also known as Java ME, is a platform which is designed for particular embedded systems. In this, the target devices such as industrial controls to mobile phones and set-top boxes are present. It is seen that Java ME was earlier called as Java 2 Platform which is Micro Edition as J2ME.

As studied, the edition of Java ME was initially invented by Sun Microsystems which was made advanced by Oracle Corporation who named it as Personal Java. Earlier, the different brands of Java ME have evolved in different JSRs. During the month of December 2006, Java ME source code was licensed under GNU which was released as phoneME.

J2SE

Since Java is a dynamic programming languages used by computer programmers today, this language carries advance features with its current edition on Java 2 Platform which is the standard edition called as J2SE. This edition is mainly used for writing applets and other applications.The main advantage of J2SE edition is that it is used in development of certain Java applications that are utilised for single computers. The J2SE edition applets and several other applications allow such functions to run smoothly. In the absence of such applications, various transactions and several Internet interactions will not takes place. With this, the edition is of a great enabler of carrying web activity.

J2EE

In order to lower the costs and fast track application design and development, Java 2 Platform Enterprise Edition called as J2EE shows a component based mechanism so as to construct, develop, assemble and deploy enterprise applications. Such platform uses multitier distributed application model. It is studied that application logic is framed into parts as per the function with certain application components which makes J2EE application to be kept on various machines according to the tier present in the multitier J2EE environment as per which the application belongs.

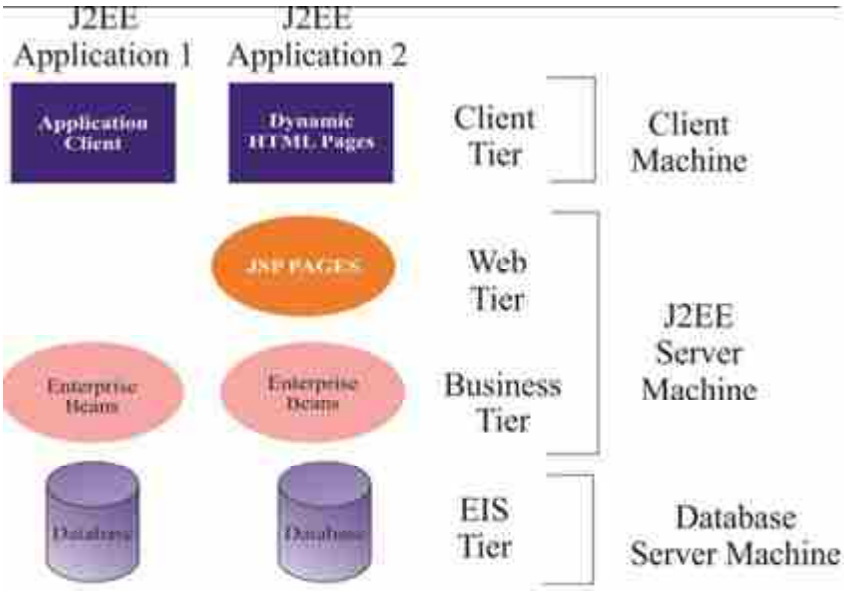


Figure 5 J2EE Multitier Applications

Figure 1.4 shows two multitier J2EE applications divided into the tiers described above. The parts shown are presented in J2EE Components as :

- Client-tier components run on the client machine
- Web-tier components mil on the J2EE server
- Business-tier components run on the J2EE server
- Enterprise information system (EIS)-tier software rurrs orr the EIS server.

❑ **Check Your Progress – 4 :**

1. Which is a multitier distributed application model ?

   a. J2ME b. J2SE c. J2EE d. all of above

2.4 Java Platform Environment :

An application runs in a platform environment, defined by the fundamental operating system, Java virtual machine, class libraries and different arrangement data complete when the application is launched.

Properties are configuration values managed as key/value pairs. In each pair, the key and value are both String values. The key identifies, and is used to retrieve, the value, much as a variable name is used to retrieve the variable's value. For example, an application capable of downloading files might use a property named "download.lastDirectory" to keep track of the directory used for the last download.

To manage properties, create instances of java.util.Properties. This class provides methods for the following :

- loading key value pairs into a Propetties object from a a stream.
- retrieving a value from its key,
- 1isting the keys and their values.
- enumerating over the keys,

  saving the properties to a stream.

Properties extend java.util.Hashtable. Some of the methods inherited from Hashtable supports following actions :

- testing to see if a particular key or value is in the Properties object.
- getting the current number of key/value pairs,
- removing a key and its value,
- adding a key/value pair to the Properties list,
- enumerating over the values or the keys,
- retrieving a value by its key,

finding out if the Properties object is empty

Benefits of 64-bit JVM on 64-bit OS and Hardware :

Wider datapath. The pipe betwen RAM and CPU is dcubled: which improves the performance of memory-bound applications.

64-bio memory addressing gives virtually unlimited (1 exabyte) heap allocation. However large heaps affect garbage collection.

Applications hat run with more than 1.5 GB of RAM should utilize the 4-bit JVM.

❑ **Check Your Progress – 5 :**

1. For an application to take place in Java, you need :

   a. JVM b. Operating system

   c. Class libraries d. all of above

2.5 Setting Path and Classpath :

It is found that there are many classes available in the library in Java. A wide range of extensive library of pre-written classes which can be applied in certain programs are listed below which are grouped into Java 1.1 packages.

- java, applet java.awt
- java.awt.data transfer java.awt.event
- java.awt.image java.awt.peer
- java.beans java.io
- java.lang java.lang.reflect
- java.math java.net
- java.rmi java.rmi.dgc
- java.rmi.registry java.rmi.server
- java.security java.security.acl
- java.security.interfaces java.sql
- java.text java.util
- java.util.zip

In this we find that every package defines a number of classes, interfaces, exceptions and errors. Also, the packages further gets splitted into sub-packages, as in case of java.lang package, which has sub-package as java.lang.reflect. It is seen that a class in sub package will not approach to a class located inside parent package. It is seen that java.net package having interfaces, classes as well as exceptions are shown below :

Interfaces in java.net

- ContentHandlerFactory
- FileXameMap
- SocketImplFactory
- URL Stream HandlerFactory

Classes in java.net

- ContentHandler
- Datagram Packet
- DatagramSocket
- DatagramSocketImpl
- HttpURLConnection
- InetAddress
- MulticastSocket
- ServerSocket
- Socket
- SocketImpl
- URL
- URLConnection
- URLEncoder
- URLStreamHandler

Exceptions in java.net

- BindException
- ConnectException
- MalfarmedURLException
- NoRouteToHostException
- ProtocolException
- SocketException
- UnknownHostException
- UnknownServiceException

❑ **Check Your Progress – 6 :**

1. In _____ package, gc() method is present.

   a. java.lang b. java.util c. java.awt d. java.io

2.6 Let Us Sum Up :

In this unit we have learnt that the Java Platform Micro Edition which is also known as Java ME, is a platform which is designed for particular embedded systems.

It is seen that Java 2 Platform Enterprise Edition also called as J2EE shows a component based mechanism so as to construct, develop, assemble and deploy enterprise applications.

An application runs in a platform environment, defined by the fundamental operating system, Java virtual machine, class libraries and different arrangement data complete when the application is launched.

2.7 Answer for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (b)

❑ **Check Your Progress 4 :**

1. (c)

❑ **Check Your Progress 5 :**

1. (d)

❑ **Check Your Progress 6 :**

1. ( )

2.8 Glossary :

1. **JVM :** It is a program called as Java virtual machine that calculates assembled Java binary codes.

2. **Class Library :** It is a collection of prewritten classes or codes in shape of templates that can be specified and used by programmer during development of an application program.

2.9 Assignment :

Find out what are the new versions of jsdk and their features.

2.10 Activities :

1. Download the latest version of java.
2. Install java.
3. Set the classpath for installed java

2.11 Case Study :

Discuss about object oriented programming languages and its architecture.

2.12 Further Readings :

1. A Primordial Interface by Wm. Paul Rogers
2. Java Diamonds Forever by Tony Sintes
3. Java Routines by John D. Mitchell
4. Object-Oriented Design and Programming by Tony Sintes

Unit

03 INTRODUCTION TO SWING

UNIT STRUCTURE

3.0 Learning Objectives :

After learning this Unit, you will be able to :

- Writing a Swing Program
- Swing Component and Containment Hierarchy

3.1 Introduction :

We see that an original Java GUI subsystem was Abstract Window Toolkit which translates visual components into platform specific where the look and feel of a component was defined by platform

Swing was introduced in 1997 with an idea to fix problems with AWT and offers two key features:

- Swii&; compcn?ncs an? 1 ighfwieigftt and don"; rely on peers
- Swing supp o ns piuggabl ? I oc k and f? ?i such as Metal i,d?faui L ; r frittdows and lvloti f. Swing ts built on .AWT

3.2 Differences between Swing and Applets :

Swing is set of platform independent UI tools which guarantees about user interface design that looks similar on several platforms. An applet is an application which runs inside browser or other hosted environment which uses Swing UI. Further an applet results as small part written in Java having plugged into HTML page. There are certain difference that exists among Swing and applets which are shown below :

Swing	Applet
Swing is light weight Component.	Applet is heavy weight Component.
Swing have look and feel according to user view you can change look and feel using UlManager.	Applet Does not provide this facility.
Swing user for stand lone Applications, Swing have main method to execute the program.	Applet need HTML code for Run the Applet.
Swing uses MVC Model view Controller.	Applet not.
Swing have its own Layout like most popular Box Layout.	Applet uses AWT Layouts like flowlayout.
Swing have some Thread rules.	Applet doesn't have any rule.
To execute Swing no need any browser By which we can create stand alone application But Here we have to add container and maintain all action control with in frame container.	To execute Applet programe we should need any one browser like Appletviewer, web broswer. Because Applet using brower container to run and all action control with in browser container.

❑ **Check Your Progress – 1 :**

1. Which among the following information is incorrect about Swing :

   a. It carries heavy weight components

   b. It carries lighter weight components

   c. It requires faster creation time

   d. it has modern design

3.3 Writing a Swing Program :

In Java, we see that there are two standard libraries for graphical user interface, where first is java.awt package having classes for windows (java.awt.Window), buttons (java.awt.Button), textfields (java.awt.TextField) and so on. Such classes are simple wrappers to platform's GUI objects where window will create along with awt in Java to look as Windows window in Windows. Consider the first example showing basic window on screen:

```
impport javax.swing;_JFrume;
import javax.swing.SwingUtilities;
public class Example extends JFrame {
  public Example() {
    setTitle("Simple example");
    setSize(300, 200);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
  }
```

```
    public static void main(String[] args) {
      Example ex = new Example();
      ex.setVisible(true);
    }
}
j
```

Being the code as very small, though application window can able to resize, maximized and minimized windows.

import javax.swing.JFrame;

import javax.swing.SwingUtilities;

Now we will import Swing classes in code example as :

public class Example extends JFrame {

Here the Example class will inherit from JFrame widget which is a toplevel container applied for keeping other widgets.

setTitle("Simple example");

Now we will set title of window using setTitle() method.

setSize(100, 100);

This will resize the window as 100px wide and 100px tall.

setLocationRelativeTo(null);

This will center the window on screen.

setDefaultCloseOperation(EXIT_ON_CLOSE);

This method will close the window, if we click on the close button of the titlebar.

By default nothing happens.

Example ex = new Example();

ex.setVisible(true);

Now we will develop an instance of code example and make it visible on screen. Here the main method is static, hence on calling, there appears no object Example. Again, main is like an external global method. Only when we explicitly create an instance (with new Example()) that an object Example, thus a JFrame appears.

Clicking on the button will terminate the application.

```
  import java.awt.event.ActionEvent;
  import java.awt.event.ActionListener;
  import java.swing.JButton;
  import java.swing.JFrame;
  import java.swing.JPanel;
  import java.swing.SwingUtilities;
  public class Example extends JFrame {
    public Example() {
      initUI();
```

```
    }
    public final void initUI() {
        JPanel panel = new JPanel();
        getContentPane().add(panel);
        panel.setLayout(null);
        JButton aquitButton = new JButton("Quit");
        quitButton.setBounds(50, 60, 80, 30);
        quitButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent event) {
            System.exit(0);
          }
        });
        panel.add(quitButton);
        setTitle("Quit button");
        setSize(300, 200);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        public void run() {
          Example ex = new Example();
          ex.setVisible(true);
        }
      }
    }
```

No we will place JButton on the window and add action listener to it.

```
  public Example() {
    initUI();
  }
```

Normally it is good practice of placing code which creates GUI inside a particular method.

JPanel panel = new JPanel();

getContentPane().add(panel);

On creating a JPanel component, we see that it is generic lightweight container that is added to JPanel to JFrame.

panel.setLayout(null);

Normally, JPanel has FlowLayout manager that is applied to place widgets on the containers. On calling setLayout(null), we will position components absolutely by using setBounds() method.

```
JButton quiButton = new JButton("Quit");
quitButton.seBonnds(50, 60, 80, 30);
quitButton.addActiionListener(new ActionListener() {
  public void actionPerformed(ActionEvent event) {
    System.exit(0);
    }
  });
```

Here we create a button by placing it by calling setBounds() method and adding an action listener. The action listener will be called, when we perform an action on the button. In our case, if we click on the button. The click will terminate the application.

```
panel.add(quitButton);
```

In order to show quit button, it must be added to panel where Swing describes small rectangular window on hovering over with mouse over an object.

```
import java.swing.JButton;
import java.swing.JFrame;
import java.swing.JPanel;

import java.swing.SwingUtilities;
public class Example extends JFrame {
  public Example() {
    initUI();
  }
public final void initUI() {
    JPanel panel = new JPanel();
    getContentPane().add(panel);
    panel.setLayout(null);
    panel.setToolTipText("A Panel container");
    JButton aquitButton = new JButton("Quit");
    button.setBounds(100, 60, 100, 30);
    button.setToolTipText("A button component");
panel.add(button);
    setTitle("Tooltip");
    setSize(300, 200);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
  }
```

```
    public static void main(String[] args) {
        public void run() {
          Example ex = new Example();
          ex.setVisible(true);

        }
      }
    }
```

Here we see that we set tooltip for frame and button.

panel.setToolTipText("A Panel container");

❑ **Check Your Progress – 2 :**

1. AWT is :

   a. Abstract windows toolkit b. Abstract Writing Toolkit

   c. Abstract win table d. none of above

2. Which of the following tool used to execute java code.

   a. javac b. rmic c. javadoc d. java

3.4 Swing Component and Containment Hierarchy :

In Java, we see that every Swing GUI applications and applets uses containment hierarchy which is not related to Swing components position on class hierarchy which can be :

Top-level Container(s)

On screen, all GUI component acts as part of containment hierarchy where one containment hierarchy is there in all program which uses Swing components. Each containment hierarchy has a top-level container at its root.

Intermediate Container(s)

It is noted that an intermediate container has content pane or panel having all visible components where all top-level container has single intermediate container if anything useful to be shown on the screen.

Atomic Component(s)

We see that button and label are atomic components that are self-sufficient entities showing bits of information to user. Moreover, atomic components gets input from user.

Top-level Containers

This containers exist mainly to show place for other Swing components to paint themselves. Swing provides four top-level container classes :

- JApplet - Enable applets to use Swing components.
- JDialog - The main class for creating a dialog window.
- JFrame - A top-level window with a title and a border.

  JWindow - As a rule, not very useful. Provides a window with no controls or title

Top-level Containers in Applets

A Swing-based applet has at least one containment hierarchy, exactly one of which is rooted by a JApplet object. In an applet which brings dialog has two containment hierarchies, components in browser window and dialog rooted by a JDialog object.

Top-level Containers in Applications

As a rule, a standalone application with a Swing-based GUI has at least one containment hierarchy with a JFrame as its root. For example, if an application has one main window and two dialogs, then the application has three containment hierarchies, and thus three top-level containers. One containment hierarchy has a JFrame as its root, and each of the other two has a JDialog object as its root.

Intermediate Containers

A panel, or pane, such as JPanel, is an intermediate container. Its only purpose is to simplify the positioning atomic components like buttons and labels. Other intermediate Swing containers, such as scroll panes (JScrollPane) and tabbed panes (JTabbedPane), typically play a more visible, interactive role in a program's GUI.

❑ **Check Your Progress – 3 :**

1. Which container has title bar and MenuBars along with button, textfield etc. ?

   a. Panel b. Frame c. Window d. Container

3.5 Let Us Sum Up :

While studying this unit, we have learnt that applet is small component written in Java which plugs into HTML page or markup language page.

Swing is a framework for developing the graphical user interface (GUI) for Java applications and applets. It is a set of platform independent UI tools (JButton, JScrollBar, etc.).

It is noted that Java offers standard libraries for graphical user interface (GUI) as java.awt package having classes to create windows (java.awt.Window), buttons (java.awt.Button), textfields (java.awt.TextField), and so on.

3.6 Answer for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (a)

❑ **Check Your Progress 2 :**

1. (a), **2.** (d0

❑ **Check Your Progress 3 :**

1. (b)

3.7 Glossary :

1. GUI : It is a Graphical User Interface that works with icons or indicators along with electronic devices

3.8 Assignment :

Write short note on Swings.

3.9 Activities :

Collect some information about swing components.

3.10 Case Study :

Discuss, how applets are differing from swing.

3.11 Further Readings :

1. The online Java tutorial @ http://docs.oracle.com/javase/tutorial/.
2. Paul Deitel and Harvey Deitel, "Java How to Program", 9th ed, 2011.
3. Y. Daniel Liang, "Introduction to Java Programming", 9th ed, 2012.
4. Bruce Eckel, "Thinking in Java", 4th ed, 2007

Unit
04

LAYOUT MANAGEMENT

UNIT STRUCTURE

4.0 Learning Objectives :

After learning this Unit, you will be able to :

- Layout Management
- Event Handling
- Action API

4.1 Introduction :

In Swing, the layout is managed by layout manager which directly places every components in the container. Without this, the components also gets placed by default layout manager. It is possible to layout the controls by hand but it becomes very difficult because of the following two reasons.

- It is very boring to handle a large number of controls within the container.
- Moreover, width and height information of component is not given when needs to arrange it.

4.2 Layout Management :

We see that Java provide with various layout manager that places the controls. The properties like size,shape and arrangement varies from one layout manager to other layout manager. When the size of the applet or the application window changes the size, shape and arrangement of the components also changes in response i.e. the layout managers adapt to the dimensions of appletviewer or the application window. The layout manager is associated with every Container object. Each layout manager is an object of the class that implements the LayoutManager interface.

If we are not using the layout manager, then we have to place the components using absolute values.

```
package zetcode;
import java.swing.JButton;
import java.swing.JFrame;
import java.swing.SwingUtilities;
public class AbsoluteExample extends JFrame {
  public Example() {
    initUI();
  }

public final void initUI() {
    setLayout(null);
    JButton ok = new JButton("OK")
    ok.setBounds(50, 50, 80, 25);
    Jbutton close = new JButton("Close");
    close.setBounds(150, 50, 80, 25);
    add(ok);
    add(close);

  setTitle("Absolute positioning");
    setSize(300, 250);
    setDefaultCloseOperation(JFrame EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
  }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
          public void run() {
            AbsoluteExample ex = new AbsoluteExample();
            ex.setVisible(true);

        }
    });
  }
}
```

In this example we see that there are two buttons.

setLayout(null);

We use absolute positioning by providing null to the setLayout() method.

```
ok.setBound(50, 50, 80, 25);
```

The setBounds() method positions the ok button. The parameters are the x, y location values and the width and height of the component.

❑ **Check Your Progress – 4 :**

1. The Swing Component classes used to Encapsulates mutually exclusive set of buttons is :

   a. AbstractButton b. ButtonGroup

   c. JButton d. ImageIcon

4.3 Event Handling :

To design an interactive Graphical User Interfaces is a difficult task, especially for those who has no experience. Creating user interfaces with the help of toolkit is time consuming process as it does not integrate in scientific-computing work-flow during explanation of algorithms and data-flow where objects shown by GUI are likely to change.

Visual computing, where the programmer creates first a graphical interface and then writes the callbacks of the graphical objects, gives rise to a slow development cycle, as the work-flow is centered on the GUI, and not on the code.

It is seen that if you want your program to calculate geometric objects, then you have to guide computer about set of 3 numbers and to tell him how to rotate that point along given axis. Also, if you want to use sphere, then a bit more work requires your program to function which will create points, spheres, etc. It knows how to rotate them, to mirror them, to scale them. So in pure procedural programming you will have procedures to rotate, scale, mirror, each one of your objects. If you want to rotate an object you will first have to find its type, then apply the right procedure to rotate it.

In object oriented programming, new abstraction just like object carries both data and procedures which is applied and altered data. In this, the data entries are known as attributes of object and procedures methods. So with the help of object oriented programming, an object is described to rotate. A point object could be implemented in python with :

```
code snippet =0
from numpy import cos: sin
class; Point (object);
  """ 3D Point objects """
  x = 0.
  y = 0.
  z = 0.
  defrotate_z(self, theta);
    """ rotate the point around the Z axis """
    xtemp = cos(theta) * self.x + sin(theta) * self.y
    ytemp = -sin(theta) * self.x + cos(theta) * self.y
    self.x = xtemp
    self.y = ytemp
```

The above program code will form a Point class. Point objects can be created as instances of the Point class as :

```
>>> from numpy import pi
>>> p = Point()
>>> p.x = 1
>>> p.rotate_z(pi)
>>> p.x
-1.0
>>> p.y
1.2246467991473532e-16
```

To carry out objects, developer requires no knowledge about internal details of their procedures. It is seen that as long as object has rotate method, the developer knows how to rotate it.

❑ **Check Your Progress – 5 :**

1. A Graphics object is :

   a. object showing part of Frame which can be drawn.

   b. object that shows whole Frame.

   c. object that shows full monitor.

   d. object that shows graphics board.

4.4 The Action Event API :

An event is an object representing a change to a resource that was observed by an event subscription. In general, requesting events on a resource is faster and subject to higher rate limits than requesting the resource itself. Additionally, change events bubble up - listening to events on a project would include when stories are added to tasks in the project, even on subtasks.

Analytics API is composed of the People API and the Events API. These APIs are used for tracking people and the events or actions they do. For instance, tracking when someone is active on your website, when a purchase is made or when someone watches a video. They are optimized for low latency and high numbers of requests, so they do not adhere to the same REST principles our other APIs use. The main Events API endpoint is /api/track, which is used to track when someone takes an action or does something.

A semantic event which shows a component-defined action occurred which is generated by component when component-specific action occurs. The event is passed to every ActionListener object that registered to receive such events using the component's addActionListener method.

The object that implements the ActionListener interface gets this ActionEvent when the event occurs. The listener is therefore spared the details of processing individual mouse movements and mouse clicks, and can instead process a "meaningful" (semantic) event like "button pressed".

❑ **Check Your Progress – 6 :**

1. In order to apply ActionListener interface, it should be placed by class. Which among the following describes the way of doing this?

   a. Creating a new class

   b. Appling graphical component to class

   c. Anonymous inner class

   d. All of above

4.5 Let Us Sum Up :

While studying this unit, we have learnt that the layout manager directly places all components in the container. If we do not use layout manager then also the components are positioned by the default layout manager.

An event is an object representing a change to a resource that was observed by an event subscription

4.6 Answer for Check Your Progress :

❑ **Check Your Progress 4 :**

1. (b)

❑ **Check Your Progress 5 :**

1. (a)

❑ **Check Your Progress 6 :**

1. (d)

4.7 Glossary :

1. **Interface :** It is a coordination involved among computer, program and humans.

4.8 Assignment :

1. Write a short note on Layout Management.
2. Write a short note on Event Handling in java.

4.9 Activities :

Collect some information on role of layout manager.

4.10 Case Study :

Generalised the basic necessity of Event API.

4.11 Further Readings :

1. The online Java tutorial @ http://docs.oracle.com/javase/tutorial/.
2. Paul Deitel and Harvey Deitel, "Java How to Program", 9th ed, 2011.
3. Y. Daniel Liang, "Introduction to Java Programming", 9th ed, 2012.
4. Bruce Eckel, "Thinking in Java", 4th ed, 2007

BLOCK SUMMARY :

In this block, students have learnt and understand about the basic of Swing Component and Containment Hierarchy and Action Event API. The block gives an idea on the study and concept of differences which exists among Swing and Applets. The students have be well explained on the concepts of Java Programs and Components and Java Platform Features.

The block detailed about the basic of Javadoc Comments techniques. The concept related to Java 2 Platform Editions and History of Java Platform will also be explained to the students. The student will be demonstrated practically about Java Platform Features technique.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. What is Event Handling ?
2. Explain the Java 2 Platform Editions ?
3. Write note on Action Event API ?
4. Write short note on Java Platform Environment ?

❖ **Long Questions :**

1. Write short notes on Java Programs and Components ?
2. Write short note on Setting Path and Classpath ?
3. Write note on differences among Swing and Applets ?

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	1	2	3	4
No. of Hrs.				

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-304

OBJECT ORIENTED CONCEPTS AND PROGRAMMING – II (ADVANCE JAVA)

BLOCK 2 : JAVA DATABASE CONNECTIVITY

JAVA DATABASE CONNECTIVITY

Block Introduction :

Java Networking is a concept of connecting two or more computing devices together so that we can share resources. Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases. In this block, we will detail about the basic of Networking Protocols and features about TCP and UDP. The block will focus on the study and concept of finding Networking Classes in JDK along with concept of socket programming. The students will give an idea on creating Client and Server program with the help of Sockets. In this block, the student will made to learn and understand about the basic of Sockets and Port numbers in that appears in Java Networking. The concept related to working and implementing of Client Server Programs along with arrangement of port numbers are also explained to students. The student will be demonstrated practically about programmed input output technique.

Block Objectives :

After learning this block, you will be able to understand :

- Understand about Networking Protocols
- Features about TCP and UDP
- Identify Networking Classes In JDK
- Concept of Sockets and Port numbers in Networking
- Qualities of Creating Client and Server Program using Sockets
- Idea about Running Client Server Programs
- Explain the various steps of writing JDBC Program
- Highlight the various JDBC Statements
- Characteristics of Prepared Statements
- Calling a Stored Procedure from JDBC

Block Structure :

Unit 5 : Networking

Unit 6 : Java Database Connectivity

Unit 7 : XML

Unit
05

NETWORKING

UNIT STRUCTURE

5.0 Learning Objectives :

After learning this Unit, you will be able to :

- Define various Networking Terminology and Protocols
- Differentiate among TCP and UDP
- Write TCP and UDP server and client program

5.1 Introduction :

The Internet is all about connecting machines together. One of the most exciting aspects of Java is that it incorporates an easy-to-use, cross-platform model for network communications. Through Java Networking we can connect two or more computing devices together so that we can share resources. Java program will communicates over the network at the application layer. Java.net package contains all the useful Java networking related classes and interfaces. Java language enables to communicate with remote file systems using a client/ server model. A server listens for connection requests from clients across the network or even from the same machine. Clients know how to connect to the server via an IP address and port number. After establishing connection, the server reads the request sent by the client and responds appropriately. In this way, applications can be broken down into specific tasks that are accomplished in separate locations. The data that is sent back and forth over a socket can

be anything. Normally, the client sends a request for information or processing to the server, which performs a task or sends data back.

5.2 Networking Terminology :

The widely used java networking terminologies are given below:

- **IP Address :** The IP address is a unique number assigned to a node of a network e.g. 192.168.200.1. It consists of octets that range from 0 to 255. The IP address can be changed. For example, each time you access the network, your device will be assigned a new IP address by the server through DHCP (dynamic host configuration protocol). This address will be used to route your data through the network from the source device to the desired destination. It exists at the network layer. IP address is divided in to classes from A to E. The IP address 127.0.0.1 is special, and is reserved to represent the loopback or localhost address.
- **Protocol :** Protocol defines the rules and conventions for communication between network devices, including ways devices can identify and make connections with each other. It is a set of rules followed for communication. Examples of protocol are TCP, FTP, Telnet, SMTP, POP etc.
- **Port Number :** The port number uniquely identifies different applications. It acts as a communication endpoint between applications. To communicate between two applications, the port number is used along with an IP Address. Port works at the transport layer. Port numbers 1 to 255 are reserved by IP for well-known services. If you connect to port 80 of a host, for instance, you may expect to find an HTTP server. On UNIX machines, ports less than 1024 are privileged and can only be bound by the root user. This is so an arbitrary user on a multi-user system can't impersonate well-known services like TELNET (port 23), creating a security problem. Windows has no such restrictions, but you should program as if it did so that your applications will work cross-platform.
- **MAC Address :** The MAC address is the physical address of the device (such as network interface card). It is fixed and each device in the world has a unique MAC address. For example, an Ethernet card may have a MAC address of 00:0c:83:b2:d0:8e. The MAC address allows computers on the same network (the same subnet) to communicate. MAC addresses works at the data-link layer.
- **Connection-Oriented and Connection-Less Protocol :** In the connection-oriented protocol, acknowledgment is sent by the receiver. So it is reliable but slow. The example of a connection-oriented protocol is TCP. But, in the connection-less protocol, acknowledgment is not sent by the receiver. So it is not reliable but fast. The example of a connection-less protocol is UDP.
- **Socket :** A socket in Java is one endpoint of a two-way communication link between two programs running on the network. A socket is an endpoint of a communication between two programs running on a network. Socket classes are used to create a connection between a client program and a server program. A socket is bound to a port number so that the TCP layer can identify the application where the data is meant to be sent to.

Java has a reasonably easy-to-use built in networking API which makes it easy to communicate via TCP/IP sockets or UDP sockets over the internet. After the connections are established, communication takes place using I/O streams. Each socket has both an OutputStream and an InputStream. The client's OutputStream is connected to the server's InputStream, and the client's InputStream is connected to the server's OutputStream.

5.3 Networking Protocols :

Networking protocols are sets of established rules that describe how to format, transmit and receive data from servers and routers to endpoints so computer network devices can communicate regardless of the differences in their underlying infrastructures, designs or standards.

To successfully send and receive information, devices on both sides of a communication must accept and follow protocol rules. Support for network protocols can be built into software, hardware or both. Standardized network protocols provide a common language for network devices. Without them, computers would not know how to communicate with each other.

Java has a rich easy-to-use built in networking API which makes it easy to communicate via TCP/IP sockets or UDP sockets over the internet. In Java, there is a java.net package provides the network support. All the classes for making a network program are defined in the java.net package. Through TCP we can communicate over the network. Typically a client opens a TCP/IP connection to a server. The client then starts to communicate with the server. When the client finishes its task, it closes the connection again.

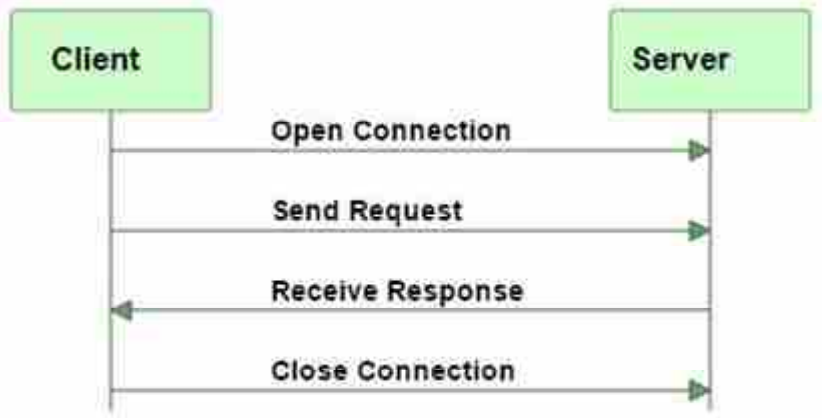


Fig 1.1 Java Client server

Client may send more than one request through an open connection. In fact, a client can send as much data as the server is ready to receive. The server can also close the connection if it wants to. When a computer (client or server) sends data to another computer over the internet it takes some time from the time the data is sent, to the data is received at the other end. This is the time it takes the data to travel over the internet. This time is called latency.

The more roundtrips you have in your protocol, the slower the protocol becomes, especially if latency is high. The HTTP protocol consists of only a single request and a single response to perform its service. A single roundtrip in other words. The SMTP protocol on the other hand, consists of several roundtrips between the client and the server before an email is sent.

The java.net package provides the functionality for two common protocols.

TCP (Transmission Control Protocol)

TCP is a connection based protocol that provides a reliable flow of data between two devices. This protocol provides the reliable connections between two applications so that they can communicate easily. It is a connection oriented protocol.

UDP (User Datagram Protocol)

UDP protocol sends independent packets of data, called datagram from one computer to another with no guarantee of arrival. It is connection less protocol.

If you want to start a server that listens for incoming connections from clients on some TCP port, you have to use a Java ServerSocket. When a client connects via a client socket to a server's ServerSocket, a Socket is assigned on the server to that connection. The client and server now communicate Socket-to-Socket.

❑ **Check Your Progress – 1 :**

1. Which of the following is false with respect to TCP ?

   a. Connection-oriented b. Process-to-process

   c. Transport layer protocol d. Unreliable

5.4 Differences between TCP and UDP :

UDP (User Datagram Protocol) is a connectionless protocol sitting on top of IP that provides unreliable packet delivery. It essentially provides user-level access to the low-level IP hardware. TCP (Transmission Control Protocol) is another protocol, a reliable but slower one, sitting on top of IP. TCP provides reliable, stream-oriented connections; can treat the connection like a stream/ file rather than packets.

(1) TCP is a reliable stream oriented protocol as opposed to UDP which is not reliable and based upon datagram. You cannot use the UDP for sending important messages which you can't afford to lose. Though there are some reliable protocols built over UDP e.g. TIBCO certified messaging which implement additional checks whether the message is delivered or not and then facilitate re-transmission.

(2) Speed: Since TCP is reliable and connection oriented it has lots of overhead as compared to UDP, which means TCP is slower than UDP and should not be used for transferring message where speed is critical e.g. live telecast, video or audio streaming. This is the reason UDP is popularly used in media transmission world.

(3) Data boundaries are preserved in case of UDP but not in the case of TCP because data is sent as it is as one message in case of UDP but TCP protocol can break and reassemble the data at sending and receiving end.

(4) TCP is a connection-oriented protocol but UDP is a connectionless protocol. What this mean is, before sending a message a connection is established between sender and receiver in TCP but no connection exists between the sender and receiver in UDP protocol.

(5) TCP provides you order guarantee but UDP doesn't provide any ordering guarantee. For example, if Sender sends 3 messages than the receiver will receive those three messages in the same order, Sender, has sent, even if they are received at different order at receiver end TCP will ensure they are delivered to a client in the order they are sent by the sender. UDP doesn't provide this feature, which means it's possible for the last message to be received first and vice-versa.

(6) TCP header size is larger than UDP header size due to excessive metadata information sent by TCP protocol. Those are required to ensure the guarantee provided by TCP protocol e.g. guaranteed ordered delivery.

(7) Multicast can only be used with UDP, it's not possible with TCP because it's a connection oriented protocol.

(8) TCP is used by HTTP, HTTPs, FTP, SMTP, Telnet protocols while UDP is used by DNS, DHCP, TFTP, SNMP, RIP, VOIP protocol.

❑ **Check Your Progress – 2 :**

1. UDP and TCP are both …… layer protocols.

   a. data link b. network c. transport d. interface

5.5 Networking Classes in java.net Package :

The java.net package contains the classes and interfaces required for networking. Some important classes are MulticastSocket, ContentHandler, URLServerSocket, Socket, InetAddress, URLConnection, DatagramSocket, and DatagramPacket. Some important interfaces in the java.net package are ContentHandlerFactory, SocketImplFactory, FileNameMap, URLStreamHandler Factory, and SocketOptions.

InetAddress Class :

This class encapsulates the numerical IP address and the domain name for the address. Factory methods of a class allow you to call the method without referencing the object. The factory methods of this class are:

- getLocalHost() method: It returns the name of the local computer
- getByName() method: It returns the address by the Domain name
- getAllByName() method: It returns all the addresses by their domain name

The instance methods of a class are methods that can be called from an object only. The instance methods for the class are:

- getAddress() method: It returns a four-element byte array that represents the object's IP address in network byte order
- getHostAddress() method: It returns the host address
- getHostName() method: It returns the hostname that is associated with the host address.

Socket Class :

Socket is a listener through which computer can receive requests and responses. Every server or programs executes on the different systems that has a socket and is bound to the specific port number. Socket provides an endpoint of two way communication link using TCP protocol. Java socket can be connection oriented or connection less. TCP provides two way communication means data can be sent across both the sides at same time.

The java.net.Socket class is used to create a socket so that both the client and the server can communicate with each other easily. A socket is an endpoint for communication between two computers. The Socket class inherits the Object class and implements the Closeable interface.

Constructors :

Constructor	Description
Socket()	It creates an unconnected socket, with the system-default type of SocketImpl.
public Socket(InetAddress address, int port)	It creates a stream socket with specified IP address to the specified port number.
public Socket(InetAddress host, int port, boolean stream)	It uses the DatagramSocket.
public Socket(InetAddress address, int port, InetAddress localAddr, int local port)	It creates a connection with specified remote address and remote port.
public Socket(Proxy, proxy)	It creates a connectionless socket specifying the type of proxy.
protected Socket(SocketImpl impl)	It creates a connectionless Socket with a user-specified SocketImpl.

Methods :

Method	Description
public InputStream getInputStream()	It returns the InputStream attached with this socket.
public OutputStream getOutput Stream()	It returns the OutputStream attached with this socket.
public synchronized void close()	It closes this socket
InetAddress getInetAddress()	It returns the InetAddress that is associated with the socket object.
int getPort()	It returns the port number on which the socket is connected
int getLocalPort()	It returns the local port number on which the socket is created

ServerSocket Class :

Socket class is used to create socket and send the request to the server. Java ServerSocket class waits for request to come over the network. It works on the basis of request and then returns a result to the request. It implements the Closeable interface.

Constructors :

Constructor	Description
ServerSocket()	It creates an unbound server socket.
ServerSocket(int port)	It creates a server socket, bound to the specified port.
ServerSocket(int port, int backlog)	It creates a server socket, bound to the specified port, with specified local port.
ServerSocket(int port, int backlog, inetAddress bindAddrs)	It creates a server socket, bound to specified port, listen backlog, and IP address.

Methods :

Method	Description
public Socket accept()	It returns the socket and establishes a connection between server and client.
public synchronized void close()	It closes the server socket.
int getLocalPort()	It returns the port number on which the server socket is listening.

DatagramPacket and DatagramSocket are the two main classes that are used to implement a UDP client/server application. DatagramPacket is a data container and DatagramSocket is a mechanism to send and receive DatagramPackets.

DatagramPacket

In UDP's terms, data transferred is encapsulated in a unit called datagram. A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed. The java.net.DatagramPacket class represents a datagram packet. They are used to implement a connectionless packet delivery service. We can create a DatagramPacket object by using one of the following constructors :

Constructors :

Constructor	Description
DatagramPacket(byte[ ] buf, int length)	It constructs a DatagramPacket for receiving packets of length.
DatagramPacket(byte[ ] buf, int length, InetAddress address, int port)	It constructs a datagram packet for sending packets of length to the specified port number on the specified host.
DatagramPacket(byte[ ] buf, int offset, int length)	It constructs a DatagramPacket for receiving packets of length, specifying an offset into the buffer.

DatagramPacket(byte[ ] buf, int offset, int length, InetAddress address, int port)	It constructs a datagram packet for sending packets of length with offset to the specified port number on the specified host.
DatagramPacket(byte[ ] buf, int offset, int length, SocketAddress address)	It constructs a datagram packet for sending packets of length with offset to the specified port number on the specified host.
DatagramPacket(byte[ ] buf, int length, SocketAddress address)	It constructs a datagram packet for sending packets of length to the specified port number on the specified host.

Methods :

Method	Description
InetAddress getAddress()	This method returns the IP address of the machine to which this datagram is being sent or from which the datagram was received.
byte[ ] getData()	This method returns the data buffer.
int getLength()	This method returns the length of the data to be sent or the length of the data received.
int getOffset()	This method returns the offset of the data to be sent or the offset of the data received.
int getPort()	This method returns the port number on the remote host to which this datagram is being sent or from which the datagram was received.
SocketAddress getSocketAddress()	This method gets the SocketAddress (usually IP address + port number) of the remote host that this packet is being sent to or is coming from.
void setAddress(InetAddress iaddr)	This method sets the IP address of the machine to which this datagram is being sent.
void setData(byte[] buf)	This method sets the data buffer for this packet.
void setData(byte[ ] buf, int offset, int length)	This method sets the data buffer for this packet.
void setLength(int length)	This method sets the length for this packet.
void setPort(int iport)	This method sets the port number on the remote host to which this datagram is being sent.

void setSocketAddress(SocketAddress address)	This method sets the SocketAddress (usually IP address + port number) of the remote host to which this datagram is being sent.

DatagramSocket

We use DatagramSocket to send and receive DatagramPackets. DatagramSocket represents a UDP connection between two computers in a network. We use DatagramSocket for both client and server. There are no separate classes for client and server like TCP sockets.

So we can create a DatagramSocket object to establish a UDP connection for sending and receiving datagram, by using one of the following constructors :

Constructors :

Constructor	Description
DatagramSocket()	It constructs a datagram socket and binds it to any available port on the local host machine.
DatagramSocket(int port)	It Constructs a datagram socket and binds it to the specified port on the local host machine.
DatagramSocket(int port, InetAddress laddr)	It creates a datagram socket, bound to the specified local address.

These constructors can throw SocketException if the socket could not be opened, or the socket could not bind to the specified port or address. So we have catch or re-throw this checked exception.

Methods :

Method	Description
Void close()	It closes this datagram socket.
InetAddress getLocalAddress()	It gets the local address to which the socket is bound.
Int getLocalPort()	It returns the port number on the local host to which this socket is bound.
Int getReceiveBufferSize()	It get value of the SO_RCVBUF option for this socket, that is the buffer size used by the platform for input on the this Socket.
Int getSendBufferSize()	It get value of the SO_SNDBUF option for this socket, that is the buffer size used by the platform for output on the this Socket.
Int getSoTimeout()	It retrieve setting for SO_TIMEOUT. 0 returns implies that the option is disabled (i.e.

Void receive(DatagramPacket p)	It receives a datagram packet from this socket.
Void send(DatagramPacket p)	It sends a datagram packet from this socket.
Void setReceiveBufferSize(int size)	It sets the SO_RCVBUF option to the specified value for this Datagram Socket.
Void setSendBufferSize(int size)	It sets the SO_SNDBUF option to the specified value for this Datagram Socket.
Void setSoTimeout(int timeout)	It enable/disable SO_TIMEOUT with the specified timeout, in milliseconds.

These methods may throw Exception like IOException, PortUnreachable Exception, SocketTimeoutException. So we have to catch or re-throw them.

URLConnection Class :

The URLConnection class is used for accessing the attribute of remote resource.

URLConnection is the superclass of all the classes that represent a communication link between application and a URL.

Methods :

Method	Description
int getContentLength()	It returns the size in byte of content associated with resource.
String getContentType()	It returns type of content found in the resource. If the content is not available, it returns null.
long getDate()	It returns the time and date of the response.
long getExpiration()	It returns the expiry time and date of the resource. If the expiry date is unavailable, it return zero.
long getLastModified()	It returns the time and date of the last modification of the resource.
InputStream getInputStream() throws IOException()	It returns an InputStream that is linked to the resource.
String getRequestProperty(String key)	It returns the value of the named general request property for the given connection.

❑ **Check Your Progress – 3 :**

1. What happens if ServerSocket is not able to listen on the specified port ?

   a. The system exits gracefully with appropriate message

   b. The system will wait till port is free

   c. IOException is thrown when opening the socket

   d. PortOccupiedException is thrown

5.6 Client and Server Program using Socket :

Creating Server :

To create the server application, we need to create the instance of ServerSocket class. Here, we are using 1234 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

ServerSocket ss=new ServerSocket(1234);

Socket s=ss.accept(); //establishes connection and waits for the client

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

Socket s=new Socket("localhost", 1234);

Let's see a simple of Java socket programming where client sends a text and server receives and prints it.

Server Program :

```
import java.io.*;
import java.net.*;
public class MySockServer {
public static void main(String[] args){
try{
ServerSocket ss=new ServerSocket(1234);
Socket s=ss.accept();//establishes connection
DataInputStream dis=new DataInputStream(s.getInputStream());
String  str=(String)dis.readUTF();
System.out.println("Data Read is= "+str);
ss.close();
}catch(Exception e){System.out.println(e);}
}
}
```

Client Program :

```
import java.io.*;
    import java.net.*;
    public class MySockClient {
    public static void main(String[] args) {
    try{
    Socket s=new Socket("localhost",1234);
    DataOutputStream dout=new DataOutputStream(s.getOutputStream());
    dout.writeUTF("Welcome to BAOU");
    dout.flush();
    dout.close();
    s.close();
    }catch(Exception e){System.out.println(e);}
    }
}
```

❑ **Check Your Progress – 4 :**

1. Which class is used to create servers that listen for either local client or remote client programs ?

   a. ServerSockets b. httpServer

   c. httpResponse d. None of the above

5.7 Executing Client Server Programs :

First run the server program. It waits until client request comes. As soon as client sends requests, server will listen and print the data received from client.

Now, run the client program from another prompt as shown below :

As soon as the client sends the requests, server receives it and displays the message as shown below.

❑ **Check Your Progress – 5 :**

1. Which classes are used for connection-less socket programming ?

   a. DatagramSocket b. DatagramPacket

   c. Both A & B d. None of the above

5.8 Multicast Protocol :

TCP and UDP are both unicast protocols; there is one sender and one receiver. Multicast packets are a special type of UDP packets. But while UDP packets have only one destination and only one receiver, multicast packets can have an arbitrary number of receivers. Multicast is quite distinct from broadcast; with broadcast packets, every host on the network receives the packet. With multicast, only those hosts that have registered an interest in receiving the packet get it.

This is similar to the way an AWTEvent and it's listeners behave in the AWT. In the same way that an AWTEvent is sent only to registered listeners, a multicast packet is sent only to members of the multicast group. AWTEvents, however, are unicast, and must be sent individually to each listener--if there are two listeners, two events are sent. With a MulticastSocket, only one is sent and it is received by many.

MulticastSocket is a subclass of DatagramSocket which has the extended ability to join and leave multicast groups. A multicast group consists of both a multicast address and a port number. The only difference between UDP and multicast in this respect is that multicast groups are represented by Class D internet addresses. Just as there are well-known ports for network services, there are reserved, well-known multicast groups for multicast network services.

When an application subscribes to a multicast group (host/port), it receives datagrams sent by other hosts to that group, as do all other members of the group. Multiple applications may subscribe to a multicast group and port concurrently, and they will all receive group datagrams.

When an application sends a message to a multicast group, all subscribing recipients to that host and port receive the message (within the time-to-live range of the packet). The application needn't be a member of the multicast group to send messages to it.

Constructors :

1. public MulticastSocket()

   It creates a multicast socket. While using this constructor we have to explicitly set all the fields such as group address, port number etc.

   Syntax : public MulticastSocket()

2. public MulticastSocket(int port)

   It creates a multicast socket bound on the port specified.

   Syntax : public MulticastSocket(int port)

   Parameters: port number to bind this socket to

3. public MulticastSocket(SocketAddress addr)

   It creates a multicast socket and binds it to specified socket address. It creates an unbound socket if address is null.

Syntax : public MulticastSocket(SocketAddress addr)

Parameters: addr- Socket address to bind this socket to

5.9 Let Us Sum Up :

In this unit we have learnt that Java Networking appears as concept that connects two or more computing devices together in order to share resources.

It is found that Java is easy-to-use built-in networking API that is easy so as to communicate using TCP/IP sockets or UDP sockets over the internet.

User Datagram Protocol is connectionless protocol above IP that provides unreliable packet delivery and provides user-level access to low-level IP hardware.

It is seen that JDK appears as class library which makes Java as powerful. Being part of Java, is considered trustworthy. It is known that socket is one end-point of two-way communication which links among two programs that runs on network

5.10 Answer for Check Your Progress :

- **Check Your Progress 1 :**

  1. (d)

- **Check Your Progress 2 :**

  1. (c)

- **Check Your Progress 3 :**

  1. (c)

- **Check Your Progress 4 :**

  1. (a)

- **Check Your Progress 5 :**

  1. (c)

5.11 Glossary :

1. UDP is a stateless, connectionless and unreliable protocol. HTTP needs connection to be established and thus, uses TCP. Telnet is a byte stream protocol which again needs connection establishment, thus uses TCP. DNS needs request and response; it needs a protocol in which a server can answer the small queries of large number of users. As UDP is fast and stateless it is the most suitable protocol and thus, it is used in DNS querying. SMTP needs reliability and thus, uses TCP. DNS uses UDP. HTTP, Telnet and SMTP uses TCP.
2. E-mail uses SMTP as application layer protocol. TCP and UDP are two transport layer protocols. SMTP uses TCP as transport layer protocol as TCP is reliable.

5.12 Assignment :

1. Explain about Networking Protocols.
2. Discuss ServerSocket and Socket class.
3. Discuss DatagramPacket and DatagramSocket class.

5.13 Activities :

Study about Creating Client and Server Program through Sockets.

5.14 Case Study :

Study about Running Client Server Programs.

5.15 Further Readings :

1. Java: The Complete Reference, Eleventh Edition by Herbert Schildt
2. https://www.javatpoint.com/java-networking
3. https://www.tutorialspoint.com/java/java_networking.htm
4. https://www.studytonight.com/java/networking-in-java.php

Unit
06

JAVA DATABASE CONNECTIVITY

UNIT STRUCTURE

6.0 Learning Objectives :

After learning this Unit, you will be :

- Able to establish database connection
- Able to define different JDBC statements
- Able to create JDBC statements
- Able to retrieve data using Result Sets

6.1 Introduction :

JDBC is advancement for ODBC, ODBC being platform dependent had a lot of drawbacks. ODBC API was written in C, C++, Python, Core Java and as we know above languages (except Java and some part of Python) are platform dependent. Therefore to remove dependence, JDBC was developed by database vendor which consists of classes and interfaces written in Java.

Java Database Connectivity is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases. JDBC lets Java programmers connect to a database, query it or update it using SQL. Java and JDBC have an essential advantage over other database programming environments since the programs developed with this technology are platform-independent and vendor-independent. Because of its universality Java and JDBC could eventually replace proprietary database languages.

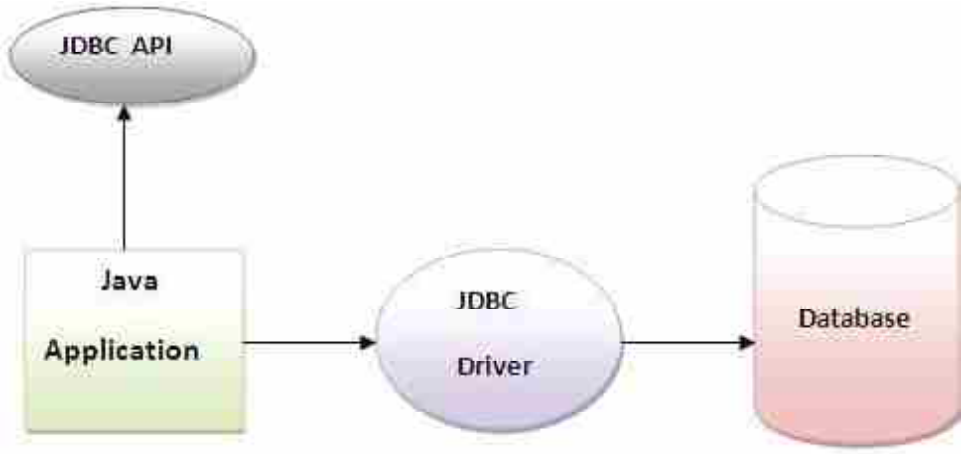


Fig 2.1 JDBC Connections

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of applications such as,

- Desktop Applications
- Java Applets
- Java Servlets
- Java ServerPages (JSPs)
- Enterprise JavaBeans (EJBs)

It is seen that Application programming interface is a document that contains description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems etc.

6.2 Types of JDBC Drivers :

The JDBC API consists of a set of interfaces and classes written in the Java programming language. Using these standard interfaces and classes, programmers can write applications that connect to databases, send queries written in structured query language (SQL), and process the results. JDBC is oriented towards relational databases.

Because JDBC is a standard specification, a Java program that uses the JDBC API can connect to any database management system (DBMS) for which there is a JDBC driver.

There are 4 different types of JDBC drivers:

Type 1 : JDBC-ODBC bridge driver

A type 1 JDBC driver consists of a Java part that translates the JDBC interface calls to ODBC calls. An ODBC bridge then calls the ODBC driver of the given database i.e. the driver converts JDBC method calls into ODBC function calls. Sun provides a JDBC-ODBC Bridge driver: sun.jdbc.odbc.JdbcOdbcDriver. This driver is native code and not Java, and is closed source.

Type 2 : Native-API Driver

A type 2 JDBC driver is like a type 1 driver, except the ODBC part is replaced with a native code part instead. The native code part is targeted at a specific database product i.e. uses the client-side libraries of the database product. The driver converts JDBC method calls into native calls of the database native API.

Type 3 : All Java + Middleware translation driver

A type 3 JDBC driver is an all Java driver that sends the JDBC interface calls to an intermediate server. The intermediate server then connects to the database on behalf of the JDBC driver. The middle-tier (application server) converts JDBC calls directly or indirectly into the vendor-specific database protocol.

Type 4 : Pure Java driver

The JDBC type 4 driver, also known as the Direct to Database Pure Java Driver, is a database driver implementation that converts JDBC calls directly into a vendor-specific database protocol. It is implemented for a specific database product. Today, most JDBC drivers are type 4 drivers.

6.3 Steps to Write a JDBC Program :

The following steps are the basic steps involve in connecting a Java application with Database using JDBC:

- Import JDBC packages.
- Load and register the JDBC driver.
- Open a connection to the database.
- Create a statement object to perform a query.
- Execute the statement object and return a query resultset
- Process the resultset
- Close the resultset and statement objects
- Close the connection

Import JDBC Packages

This is for making the JDBC API classes immediately available to the application program. The following import statement should be included in the program irrespective of the JDBC driver being used:

```
import java.sql.*;
```

Load and Register the JDBC Driver

This is for establishing a communication between the JDBC program and the database. This is done by using the static registerDriver() method of the DriverManager class of the JDBC API.

Connecting to a Database

After loading the driver, the next step is to create and establish the connection. Once required, packages are imported and drivers are loaded and registered, then we can go for establishing a Database connection. This is done by using the getConnection() method of the DriverManager class. A call to this method creates an object instance of the java.sql.Connection class. The

getConnection() requires three input parameters, namely, a connect string (url), a username and a password.

- Connection con = DriverManager.getConnection(url,user,password)

Querying the Database

Querying the database involves two steps: first, creating a statement object to perform a query, and second, executing the query and returning a resultset.

Creating a Statement Object

This is to instantiate objects that run the query against the database connected to. This is done by the createStatement() method of the Connection object. A call to this method creates an object instance of the Statement class. The following line of code illustrates this:

Statement stmt = con.createStatement();

createStatement method is defined in Connection class and used to execute the sql queries.

Executing the Query and Returning a ResultSet

Once a Statement object has been constructed, the next step is to execute the query. This is done by using the executeQuery() method of the Statement object. A call to this method takes as parameter a SQL SELECT statement and returns a JDBC ResultSet object. The following line of code illustrates this using the stmt object created earlier:

ResultSet rs = stmt.executeQuery

("SELECT studno, studname, studadd, studres FROM student ORDER BY studname");

Processing the results of a database query that returns multiple rows

Once the query has been executed, there are two steps to be carried out:

- Processing the output resultset to fetch the rows
- Retrieving the column values of the current row

The first step is done using the next() method of the ResultSet object. A call to next() is executed in a loop to fetch the rows one row at a time, with each call to next() advancing the control to the next available row. The next() method returns the Boolean value true while rows are still available for fetching and returns false when all the rows have been fetched.

The second step is done by using the getXXX() methods of the JDBC rs object. Here getXXX() corresponds to the getInt(), getString() etc with XXX being replaced by a Java datatype.

❑ **Check Your Progress – 1 :**

1. Which of the following is advantage of using PreparedStatement in Java ?

   a. Slow performance b. Encourages SQL injection

   c. Prevents SQL injection d. More memory usage

6.4 Establishing a Connection :

In order to connect any java application with database using JDBC, we have to establish a connection using following steps :

1. 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. The forName() method is valid only for JDK Compliant Virtual Machines. The syntax of forName() method is:

public static void forName(String className) throws ClassNotFoundException

// Oracle database

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

// For MySQL Database

- Class.forName("com.mysql.jdbc.Driver");

The driver's class file loads into the memory at runtime. It implicitly loads the driver. While loading, the driver will register with JDBC automatically. The following table lists the JDBC driver name for the different databases :

Database Name	JDBC Driver Name
MySQL	com.mysql.jdbc.Driver
Oracle	oracle.jdbc.driver.OracleDriver
Microsoft SQL Server	com.microsoft.sqlserver.jdbc.SQLServerDriver
MS Access	net.ucanaccess.jdbc.UcanaccessDriver
PostgreSQL	org.postgresql.Driver
IBM DB2	com.ibm.db2.jdbc.net.DB2Driver
Sybase	com.sybase.jdbcSybDriver

2. Create the connection object

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

- public static Connection getConnection(String url)throws SQLException
- public static Connection getConnection(String url,String name,String password) throws SQLException

// Oracle database

- Connection con=DriverManager.getConnection(

  "jdbc:oracle:thin:@localhost:1521:EMP","system","password");

  Where, jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, you may also use IP address, 1521 is the port number and EMP is the database name. The default username for the oracle database is system and the password is the password given by the user at the time of installing the oracle database.

// For MySQL Database

Connection con = DriverManager.getConnection("jdbc:mysql://localhost/EMP",Username,Password);

The following table lists the JDBC connection strings for the different databases :

Database	Connection String / DB URL
MySQL	jdbc:mysql://HOST_NAME:PORT/DATABASE_NAME
Oracle	jdbc:oracle:thin:@HOST_NAME:PORT:SERVICE_NAME
Microsoft SQL Server	jdbc:sqlserver://HOST_NAME:PORT;DatabaseName=< DATABASE_NAME>
MS Access	jdbc:ucanaccess://DATABASE_PATH
PostgreSQL	jdbc:postgresql://HOST_NAME:PORT/DATABASE_NAME
IBM DB2	jdbc:db2://HOSTNAME:PORT/DATABASE_NAME
Sybase	jdbc:Sybase:Tds:HOSTNAME:PORT/DATABASE_NAME

3. 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. The syntax of createStatement() method is :

```
public Statement createStatement() throws SQLException
```

4. 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. The syntax of executeQuery() method is :

```
public ResultSet executeQuery(String sql) throws SQLException
```

5. 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. The syntax of close() method is:

```
public void close()throws SQLException
```

❑ **Check Your Progress – 2 :**

1. Which of the following is method of JDBC batch process ?

   a. setBatch() b. deleteBatch()

   c. removeBatch() d. addBatch()

6.5 Creating JDBC Statements :

Once a connection is obtained we can interact with the database. The JDBC Statement, CallableStatement and PreparedStatement interfaces define the methods and properties that enable you to send SQL or PL/SQL commands and receive data from your database. They also define methods that help bridge data type differences between Java and SQL data types used in a database.

A java.sql.Connection object has several methods used to create a java.sql.Statement object, which we can use to execute SQL queries against a database.

Methods of Connection :

1. public Statement createStatement() : It creates a statement object that can be used to execute SQL queries.
2. public Statement createStatement(int resultSetType,int resultSet Concurrency) : It creates a Statement object that will generate ResultSet objects with the given type and concurrency. In Jdbc ResultSet Interface are classified into two type:
   - Non-Scrollable ResultSet in JDBC
   - Scrollable ResultSet

By default a ResultSet Interface is Non-Scrollable. In non-scrollable ResultSet we can move only in forward direction (that means from first record to last record), but not in Backward Direction. If we want to move in backward direction use Scrollable Interface.

The type and mode are predefined in ResultSet Interface of Jdbc like below which is static final.

Type :

i. public static final int TYPE_FORWARD_ONLY

ii. public static final int TYPE_SCROLL_INSENSITIVE

iii. public static final int TYPE_SCROLL_SENSITIVE

Mode :

i. public static final int CONCUR_READ_ONLY

ii. public static final int CONCUR_UPDATABLE

Example :

Statement stmt= con.CreateStatement(ResutlSet.TYPE_SCROLL_ INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

3. public void setAutoCommit(boolean status) : It is used to set the commit status.By default it is true.
4. public void commit() : It saves the changes made since the previous commit/rollback permanent.
5. public void rollback() : It drops all changes made since the previous commit/rollback.
6. public void close() : It closes the connection and Releases a JDBC resources immediately.

When we have got a java.sql.Statement object, several methods on that object can be used to execute a SQL statement against the database. We see that there are various important methods from the java.sql.Statement interface to use:

The Statement interface provides the following important methods :

Methods	Description
public boolean execute (String sql)	It executes the given SQL query, which may return multiple results.
public intexecuteBatch( )	It submits the batch of commands to the database and returns an array of update counts.

public ResultSetexecute Query( )	Executes the given SQL queries which return the single ResultSet object.
publicintexecuteUpdate (String sql)	It performs the execution of DDL (insert, update or delete) statements.
public Connection getConnection ( )	It retrieves the connection object that produced the statement object.

❑ **Check Your Progress – 3 :**

1. The interface ResultSet has a method, getMetaData(), that returns a/an

   a. Tuple b. Value c. Object d. Result

6.6 Manipulating Result Sets :

The type of a ResultSet object determines the level of its functionality in two areas: the ways in which the cursor can be manipulated, and how concurrent changes made to the underlying data source are reflected by the ResultSet object.

JDBC returns results in a ResultSet object, so we need to declare an instance of the class ResultSet to hold our results. In addition, the Statement methods executeQuery and getResultSet both return a ResultSet object, as do various DatabaseMetaData methods. The following code demonstrates declaring the ResultSet object rs and assigning the results of query to it by using the executeQuery method.

First we need to create a scrollable ResultSet object. The following line of code illustrates to create a scrollable ResultSet object:

```
Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_
SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
ResultSet rs = stmt.executeQuery("select * from student");
```

There are 4 categories of ResultSet methods. They are :

1. Navigational Methods :

This method is used to move the cursor around the dataset.

- **Boolean absolute(int row):** It is used to move the cursor to the specified row which is mentioned in the parameter and return true if the operation is successful else return false.
- **Void afterLast():** It makes the ResultSet cursor to move after the last row.
- **Void beforeFirst():** It makes the ResultSet cursor to move before the first row.
- **Boolean first():** It makes the ResultSet cursor to move to the first row. It returns True if the operation is successful else False.
- **Boolean last():** It makes the ResultSet cursor to move to the last row. It returns True if the operation is successful else False.
- **Boolean next():** It makes the ResultSet cursor to move to the next row. It returns True if there are more records and False if there are no more records.

- **Boolean previous():** It makes the ResultSet cursor to move to the previous row. It returns True if the operation is successful else False.
- **Boolean relative():** It moves the cursor to the given number of rows either in the forward or backward direction.
- **Int getRow():** It returns the current row number the ResultSet object is pointing now.
- **Void moveToCurrentRow():** It moves the cursor back to the current row if it is currently in insert row.
- **Void moveToInsertRow():** It moves the cursor to the specific row to insert the row into the Database. It remembers the current cursor location. So we can use the moveToCurrentRow() method to move the cursor to the current row after the insertion.

2. Getter Methods

ResultSet has stored the data of the table from the Database. Getter methods are used to get the values of the table in ResultSet. For that, we need to pass either column Index value or Column Name.

The following are the getter methods in ResultSet :

- **int getInt(int ColumnIndex):** It is used to get the value of the specified column Index as an int data type.
- **float getFloat(int ColumnIndex):** It is used to get the value of the specified column Index as a float data type.
- **java.sql.date getDate(int ColumnIndex):** It is used to get the value of the specified column Index as a date value.
- **int getInt(String ColumnName):** It is used to get the value of the specified column as an int data type.
- **float getFloat(String ColumnName):** It is used to get the value of the specified column as a float data type.
- **Java.sql.date getDate(String ColumnName):** It is used to get the value of the specified column as a date value.

There are getter methods for all primitive data types (Boolean, long, double) and String also in ResultSet interface.

3. Setter / Updater Methods

We can update the value in the Database using ResultSet Updater methods. It is similar to Getter methods, but here we need to pass the values/ data for the particular column to update in the Database.

The following are the updater methods in ResultSet :

- **void updateInt(int ColumnIndex, int Value):** It is used to update the value of the specified column Index with an int value.
- **void updateFloat(int ColumnIndex, float f):** It is used to update the value of the specified column Index with the float value.
- **void updateDate(int ColumnIndex, Date d):** It is used to update the value of the specified column Index with the date value.
- **void updateInt(String ColumnName, int Value):** It is used to update the value of the specified column with the given int value.

- **void updateFloat(String ColumnName, float f):** It is used to update the value of the specified column with the given float value.
- **Java.sql.date getDate(String ColumnName):** It is used to update the value of the specified column with the given date value.

There are Updater methods for all primitive data types (Boolean, long, double) and String also in ResultSet interface.

Updater methods just update the data in the ResultSet object. Values will be updated in DB after calling the insertRow or updateRow method.

Updating a Row :

We can update the data in a row by calling updateX() methods, passing the column name or index, and values to update. We can use any data type in place of X in the updateX method. Till now, we have updated the data in the ResultSet object. To update the data in DB, we have to call the updateRow() method.

Inserting a Row :

We need to use moveToInsertRow() to move the cursor to insert a new row. We have already covered this in the Navigation methods section. Next, we need to call updateX() method to add the data to the row. We should provide data for all the columns else it will use the default value of that particular column.

After updating the data, we need to call the insertRow() method. Then use the moveToCurrentRow() method, to take the cursor position back to the row we were at before we started inserting a new row.

4. Miscellaneous Methods

- **void close():** It is used to close the ResultSet instance and free up the resources associated with ResultSet instance.
- **ResultSetMetaData getMetaData():** It returns the ResultSetMetaData Instance. It has the information about the type and property of columns of the query output. We will learn more about ResultSetMetaData in the next section.

Using the Method next :

The variable rs, an instance of ResultSet, contains the rows of student table. In order to access the student records, we will go to each row and retrieve the values according to their types. The method next moves what is called a cursor to the next row and makes that row (called the current row) the one upon which we can operate. Since the cursor is initially positioned just above the first row of a ResultSet object, the first call to the method next moves the cursor to the first row and makes it the current row. Successive invocations of the method next move the cursor down one row at a time from top to bottom. Note that with the JDBC 2.0 API, covered in the next section, you can move the cursor backwards, to specific positions, and to positions relative to the current row in addition to moving the curs or forward.

Using the getXXX Methods :

We can use the getXXX method of the appropriate type to retrieve the value in each column. For example, the first column in each row of rs is studno, which stores a value of SQL type int. The method for retrieving a value of SQL type integer is getInt. The second column in each row stores a value

of student name of SQL type varchar, and the method for retrieving values of that type is getString. The following code accesses the values stored in the current row of rs and prints a line with the name followed by three spaces and the price. Each time the method next is invoked, the next row becomes the current row, and the loop continues until there are no more rows in rs.

```
String query = "SELECT * from student";
ResultSet rs = stmt.executeQuery(query);
while (rs.next()) {
int num= rs.getInt("studno");
String s=rs.getString("studname");
String add=rs.getString("studadd");
float n = rs.getFloat("studres");
System.out.println(num +" " +s + "" + add +" "+ n);
}
```

The output will look something like this :

```
1 mukesh sanand 70.5
2 ved gandhinagar 80.6
3 shrey vasai 80.8
```

Using the Method getString :

Although the method getString is recommended for retrieving the SQL types CHAR and VARCHAR, it is possible to retrieve any of the basic SQL types with it. (You cannot, however, retrieve the new SQL3 datatypes with it.) Getting all values with getString can be very useful, but it also has its limitations. For instance, if it is used to retrieve a numeric type, getString will convert the numeric value to a Java String object, and the value will have to be converted back to a numeric type before it can be operated on as a number. In cases where the value will be treated as a string anyway, there is no drawback.

❑ **Check Your Progress – 4 :**

1. Which JDBC driver Type(s) can be used in either applet or servlet code?

   a. Both Type 1 and Type 2 b. Both Type 1 and Type 3

   c. Both Type 3 and Type 4 d. Type 4 only

6.7 Using Prepared Statements :

The PreparedStatement interface extends the Statement interface. It represents precompiled SQL statements and stores it in a PreparedStatement object. It increases the performance of the application because the query is compiled only once. In the example of parameterized query, we see that:

```
String sql="insert into student values(?,?,?)";
```

In this, the parameter (?) is passed for values which will be set by calling setter methods of Prepared Statement.

The important methods of PreparedStatement interface are listed below :

Methods	Description
public void setInt(int paramIndex, int value)	It sets the integer value to the given parameter index.
public void setString(int paramIndex, String value)	It sets the String value to the given parameter index.
public void setFloat(int paramIndex, float value)	It sets the float value to the given parameter index.
public void setDouble(int paramIndex, double value)	It sets the double value to the given parameter index.
public int executeUpdate()	It executes the query. It is used for create, drop, insert, update, delete etc.
public ResultSet executeQuery()	It executes the select query. It returns an instance of ResultSet.

Example : Insert operation with PreparedStatement Interface

- Creating PreparedStatement Object

```
String sql = "Select * from student where studno= ?";
PreparedStatementps = con.prepareStatement(sql);
```

Note : All the parameter are represented by "?" symbol and each parameter is referred to by its origin position.

```
import java.sql.*;
public class PreparedStatTest
{
    public static void main(String args[])
    {
      try
      {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:baou","username",
"password");
        PreparedStatementps = con.prepareStatement("insert into student
values(?, ?, ?,?)");
        ps.setInt(1, 5);
        ps.setString(2, "Het");
        ps.setString(3, "Mumbai");
        ps.setFloat(4, 88.4);
        ps.executeUpdate();
        con.close();
    }
```

```
        catch(Exception e)
        {
          System.out.println(e);
        }
      }
    }
```

To updates the record :

```
PreparedStatement stmt=con.prepareStatement("update student set studname=? where studno=?");
stmt.setString(1,"Kalp");//1 specifies the first parameter in the query i.e. name
stmt.setInt(2,1);
int i=stmt.executeUpdate();
System.out.println(i+"records updated");
```

To delete the record :

```
PreparedStatement stmt=con.prepareStatement("delete from student where studno=?");
stmt.setInt(1,3);
int i=stmt.executeUpdate();
System.out.println(i+" records deleted");
```

❑ **Check Your Progress – 5 :**

1. Which of the following contains both date and time ?

   a. java.io.date b. java.sql.date

   c. java.util.date d. java.util.dateTime

6.8 Using Callable Statement :

A Stored procedure can return result sets, we can use getResultSet method in the CallableStatement class to retrieve return result sets. When a procedure has return value for an OUT parameter, we must tell the JDBC driver what SQL type the value will be, with the registerOutParameter method. To call stored procedures, we invoke methods in the CallableStatement class.

The basic steps are :

- Creating a CallableStatement object by calling the Connection.prepareCall method.
- Using the CallableStatement.setXXX methods to pass values to the input (IN) parameters.
- Using the CallableStatement.registerOutParameter method to indicate which parameters are output-only (OUT) parameters, or input and output (INOUT) parameters.
- Invoke one of the following methods to call the stored procedure:
  - CallableStatement.executeUpdate method, if the stored procedure does not return result sets.

- CallableStatement.executeQuery method, if the stored procedure returns one result set.
- CallableStatement.execute method, if the stored procedure returns multiple result sets.

- Calling the CallableStatement.getResultSet method to obtain the result set (which is in a ResultSet object), if the stored procedure returns one result set. But if the stored procedure returns result sets, retrieve the result sets by combining CallableStatement.getResultSet and CallableStatement.getMoreResults methods.
- Using the CallableStatement.getXXX methods to retrieve values from the OUT parameters or INOUT parameters.
- Calling the CallableStatement.close method to close the CallableStatement object when we have finished using that object.

Example :

Creating CallableStatement Interface:

The instance of a CallableStatement is created by calling prepareCall() method on a Connection object.

E.g.

CallableStatementcallableStatement = con.prepareCall("{call procedures (?,?)}");

Creating stored procedure

```
create or replace procedure "insertStudents"
(rollno IN NUMBER,
name IN VARCHAR2,
address IN VARCHAR2,
result IN NUMBER)
is
begin
    insert into students values(rollno, name, address, result);
end;
/
```

// ProcedureTest.java

```
import java.sql.*;
public class ProcedureTest
{
   public static void main(String args[])
   {
      try
      {
         Class.forName("oracle.jdbc.driver.OracleDriver");
         Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:baou","scott","tiger");
```

```
                CallableStatementstmt = con.prepareCall("{call insertStudents
(?, ?,?,?) }");
                stmt.setInt(1, 6);
                stmt.setString(2, "Vinod");
                stmt.setString(3, "Ahmedabad");
                stmt.setFloat(4, 74.36);
                stmt.execute();
                System.out.println("Record inserted successfully");
                con.close();
                stmt.close();
            }
            catch(Execption e)
            {
            e.printStackTrace();
        }
    }
}
```

Note : The ProcedureTest.java file inserts the record in Students table in Oracle database using the stored procedure.

6.9 ResultSetMetaData :

Metadata means data about data. Using this interface, we will get more information about ResultSet. It is available in the java.sql package. Every ResultSet object is associated with one ResultSetMetaData object. This object will have the details of the properties of the columns like datatype of the column, column name, number of columns, table name, schema name, etc., We can get the ResultSetMetaData object using the getMetaData() method of ResultSet. Syntax of the ResultSetMetaData is:

```
PreparedStatement pst1 = conn.prepareStatement(insert_query);
ResultSet rs1 = pst1.executeQuery("Select * from STUDENT_MASTER");
ResultSetMetaData rsmd = rs.getMetaData();
```

Important methods of ResultSetMetaData interface are :

Method Name	Description
String getColumnName(int column)	It returns the column name of the particular column
String getColumnTypeName(int column)	It returns the datatype of the particular column which we have passed as a parameter
String getTableName(int column)	It returns the table name of the column
String getSchemaName(int column)	It returns the schema name of the column's table
int getColumnCount()	It returns the number of columns of the ResultSet

boolean isAutoIncrement(int Column)	It returns true if the given column is Auto Increment, else false
boolean isCaseSensitive(int Column)	It returns true if the given Column is Case Sensitive, else false

6.10 Let Us Sum Up :

While studying this unit, we have learnt that Java Database Connectivity, is a standard Java API for database-independent connectivity that exists among Java programming language and wide databases.

JDBC has two levels of interface where main interface carries an API from JDBC manager which communicates with individual database product drivers, JDBC-ODBC bridge and JDBC network driver when Java program is running in network environment.

Querying the database involves creating a statement object to perform a query and executing the query and returning a resultset.

It is noted that once connection is obtained interaction with database appears where JDBC Statement, Callable Statement and Prepared Statement define methods and properties which sends SQL or PL/SQL commands and receive data from database.

A Stored procedure can return result sets, you can use getResultSet method in the CallableStatement class to retrieve return result sets.

6.11 Answer for Check Your Progress :

- **Check Your Progress 1 :**

  1. (c)

- **Check Your Progress 2 :**

  1. (d)

- **Check Your Progress 3 :**

  1. (c)

- **Check Your Progress 4 :**

  1. (c)

- **Check Your Progress 5 :**

  1. (d)

6.12 Glossary :

1. **Java Database Connectivity :** Standard Java API for database that involves Java programming language with databases.
2. **Prepared Statement :** Statement used to find parameterized query.
3. **DriverManager :** This class manages a list of database drivers. It Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.
4. **Driver :** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead,

you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.

5. **Connection :** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
6. **Statement :** You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
7. **ResultSet :** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
8. **SQLException :** This class handles any errors that occur in a database application.

6.13 Assignment :

1. Write short note on establishing a connection with mysql database.
2. Explain the concept of Callable Statement using example.

6.14 Activities :

Collect important methods of PreparedStatements and ResultSetMetadata.

6.15 Case Study :

Generalise the basics of Java Database Connectivity.

6.16 Further Readings :

1. Java: The Complete Reference, Eleventh Edition by Herbert Schildt
2. https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html
3. https://www.javatpoint.com/java-jdbc

Unit
07

XML

UNIT STRUCTURE

7.0 Learning Objectives :

After learning this Unit, you will be :

- Able to define XML
- Able to write XML file
- Able to create DTD and XML Schema
- Able to define parser and XML Namespace

7.1 Introduction :

XML stands for EXtensible Markup Language. It is a markup language much like HTML. It was designed to describe data, not to display data. XML tags are not predefined. You must define your own tags. It is designed to be self-descriptive. It is a W3C Recommendation. It is not a replacement for HTML. It is a complement to HTML. It is a software and hardware independent tool for carrying information.

XML and HTML were designed with different goals :

- XML was designed to describe data, with focus on what data is
- HTML was designed to display data, with focus on how data looks
- HTML is about displaying information, while XML is about carrying information.

The first line in an XML document is called the prolog.

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

The prolog is optional. Normally it contains the XML version number. It can also contain information about the encoding used in the document. This prolog specifies UTF-8 encoding:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The XML standard states that all XML software must understand both UTF-8 and UTF-16. UTF-8 is the default for documents without encoding information. In addition, most XML software systems understand encodings like ISO-8859-1, Windows-1252, and ASCII.

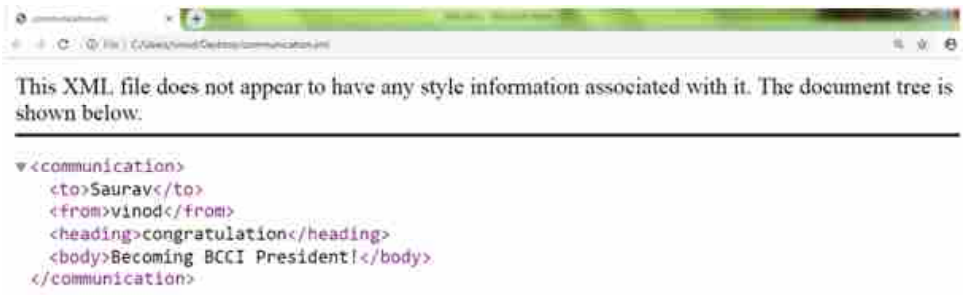
An example of XML Document :

XML documents use a self-describing and simple syntax. XML file is saved with xml extension. For example, the following is a complete XML file presenting a communication.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<communication>
    <to>Saurav</to>
    <from>vinod</from>
    <heading>congratulation</heading>
    <body>Becoming BCCI President!</body>
</communication>
```

To view an XML document in browser you can click on a link, type the URL in the address bar, or double-click on the name of an XML file in a files folder. Once opened, it will display the document with color coded root and child elements. Depending on the browser a symbol (sign) to the left of the elements can be clicked to expand or collapse the element structure. If you want to view the raw XML source, you must select "View Source" from the browser menu. If an erroneous XML file is opened, the browser will report the error. Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table. Without any information about how to display the data, most browsers will just display the XML document as it is.

The output of the above example will look like below in chrome browser.



7.2 Application of XML :

XML is used in many aspects of web development, often to simplify data storage and sharing.

- **XML Separates Data from HTML**

If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes. With XML,

data can be stored in separate XML files. This way you can concentrate on using HTML/CSS for display and layout, and be sure that changes in the underlying data will not require any changes to the HTML. With a few lines of JavaScript code, you can read an external XML file and update the data content of your web page.

- **XML Simplifies Data Sharing**

In the real world, computer systems and databases contain data in incompatible formats. XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data. This makes it much easier to create data that can be shared by different applications.

- **XML Simplifies Data Transport**

One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet. Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

- **XML Simplifies Platform Changes**

Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost. XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

- **XML Makes Your Data More Available**

Different applications can access your data, not only in HTML pages, but also from XML data sources. With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc.), and make it more available for blind people, or people with other disabilities.

- **Internet Languages Written in XML**

Many Internet languages are written in XML. Some examples are :

- XHTML
- XML Schema
- SVG
- WSDL

7.3 Well Formed and Valid XML Documents :

XML documents that conform to the following syntax rules are said to be "Well Formed" XML documents. It is also known as syntax rules of XML.

- **All XML elements must have a closing tag**

In HTML, some elements do not have to have a closing tag:

Welcome to India.

In XML, it is illegal to omit the closing tag. All elements must have a closing tag:

 Welcome to India.

- **XML tags are case sensitive**

XML tags are case sensitive. The tag <Name> is different from the tag <name>.

Opening and closing tags must be written with the same case:

<Name>This is incorrect</name>

<Name>This is correct</Name>

- **XML elements must be properly nested**

In HTML, you might see improperly nested elements:

This text is bold and italic

In XML, all elements must be properly nested within each other:

This text is bold and italic

In the above example, "Properly nested" simply means that since the element is opened inside the element, it must be closed inside the element.

- **XML documents must have a root element**

XML documents must contain one element that is the parent of all other elements. This element is called the root element.

<root> <child> <subchild>.....</subchild> </child> </root>

- **XML attribute values must be quoted**

XML elements can have attributes in name/value pairs just like in HTML. In XML, the attribute values must always be quoted.

Incorrect :

<note date=07/02/2021>

<to>saurav</to>

<from>vinod</from> </note>

Correct:

<note date="12/11/2007">

<to>saurav</to>

<from>vinod</from> </note>

The error in the first document is that the date attribute in the note element is not quoted.

- **Entity References**

Some characters have a special meaning in XML. If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

<message>if salary < 5000 then</message>

To avoid this error, replace the "<" character with an entity reference:

<message>if salary < 5000 then</message>

There are 5 pre-defined entity references in XML :

&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark
&lt;	<	less than

- **Comments in XML**

The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment -->

- **White-space is preserved in XML**

XML does not truncate multiple white-spaces in a document (while HTML truncates multiple white-spaces to one single white-space):

XML: Welcome Saurav

HTML: Welcome Saurav

- **XML Stores New Line as LF**

Windows applications store a new line as: carriage return and line feed (CR+LF). Unix and Mac OSX uses LF. Old Mac systems uses CR. XML stores a new line as LF.

A "valid" XML document is not the same as a "well formed" XML document. A valid XML document must be well formed. In addition it must conform to a document type definition.

Rules that defines the legal elements and attributes for XML documents are called Document Type Definitions (DTD) or XML Schemas. There are two different document type definitions that can be used with XML:

- DTD : The original Document Type Definition
- XML Schema : An XML-based alternative to DTD

With a DTD, independent groups of people can agree to use a standard DTD for interchanging data. Your application can use a standard DTD to verify that the data you receive from the outside world is valid.

❑ **Check Your Progress – 1 :**

1. What does XML stand for ?

a. eXtra Modern Link b. eXtensible Markup Language

c. Example Markup Language d. X-Markup Language

7.4 XML Namespace :

XML Namespaces provide a method to avoid element name conflicts. In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

Following XML carries HTML table information :

<table>

<tr>

<td>Bharat</td>

<td>Gujarat</td>

</tr>

</table>

This XML carries information about a table (Gujarat data):

<table>

<name>Gujarat</name>

<capital>gandhinagar</capital>

</table>

If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning. A user or an XML application will not know how to handle these differences. Name conflicts in XML can easily be avoided using a name prefix.

Following XML carries information about an HTML table, and Gujarat data:

<h:table>

<h:tr>

<h:td>Bharat</h:td>

<h:td>Gujarat</h:td>

</h:tr>

</h:table>

<g:table>

<g:name>Gujarat</g:name>

<g:capital>gandhinagar</g:capital>

</g:table>

In the example above, there will be no conflict because the two <table> elements have different names.

- **XML Namespaces : The xmlns Attribute**

When using prefixes in XML, a so-called namespace for the prefix must be defined. The namespace is defined by the xmlns attribute in the start tag of an element. The namespace declaration has the following syntax:

xmlns:prefix="URI".

<root>

<h:table xmlns:h="http://www.w3.org/TR/">

<h:tr>

<h:td>Bharat</h:td>

<h:td>Gujarat</h:td>

</h:tr>

</h:table>

Here, the Namespace prefix is h, and the Namespace identifier (URI) as http://www.w3.org/TR/. This means, the element names and attribute names with the h prefix all belong to the http://www.w3.org/TR/ namespace.

```
<g:table xmlns:g="http://www.w3.org/gujarat">
<g:name>Gujarat</g:name>
<g:capital>gandhinagar</g:capital>
</g:table>
</root>
```

Here, the Namespace prefix is g, and the Namespace identifier (URI) as http://www.w3.org/gujarat. This means, the element names and attribute names with the g prefix all belong to the http://www.w3.org/gujarat namespace.

When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace. Namespaces can be declared in the elements where they are used or in the XML root element:

```
<root xmlns:h="http://www.w3.org/TR/" xmlns:g="http://www.w3.org/gujarat">
<root>
<h:table>
<h:tr>
<h:td>Bharat</h:td>
<h:td>Gujarat</h:td>
</h:tr>
</h:table>
<g:table>
<g:name>Gujarat</g:name>
<g:capital>gandhinagar</g:capital>
</g:table>
</root>
```

The namespace URI is not used by the parser to look up information. The purpose is to give the namespace a unique name. However, often companies use the namespace as a pointer to a web page containing namespace information.

Another Example :

Following XML contains a list of tutorials for client side tools of web development :

```
<?xml version="1.0"?>
<tools>
<name>html</name>
<name>css</name>
<name>javascript</name>
</tools>
```

This XML contains a list of tutorials for server side tools of web development :

```
<?xml version="1.0"?>
<tools>
<name>php</name>
<name>asp</name>
<name>jsp</name>
</tools>
```

If these two XML instances are put together, since both of them carry elements with tagname name, it will create a naming collision. An XML parser will not be able to understand the meaning of using tutorial element is actually for different purposes. Using namespace, this problem can be solved :

```
<?xml version="1.0"?>
<client:tools xmlns:client ="https://www.baounamespace.com/server-side-tools">
<client:name>html</client:name>
<client:name>css</client:name>
<client:name>javascript</client:name>
</client:tools>
<server:tools xmlns:server ="https://www.baounamespace.com/server-side-tools">
<server:name>php</server:name>
<server:name>asp</server:name>
<server:name>jsp</server:name>
</ server:tools>
```

- **Default Namespaces**

Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:

```
xmlns="namespaceURI"
```

This XML carries HTML table information:

```
<table xmlns="http://www.w3.org/TR">
<tr>
<td>Bharat</td>
<td>Gujarat</td>
</tr> </table>
```

❑ **Check Your Progress – 2 :**

1. The attribute used to define a new namespace is

   a. XMLNS b. XmlNameSpace

   c. Xmlns d. XmlNs

7.5 XML Parser :

XML parser is a software library or a package that provides interface for client applications to work with XML documents. It checks for proper format of the XML document and may also validate the XML documents. First parser load the XML file and other related files. After it, parser also checks the validity of XML document and create a document tree structure. Modern day browsers have built-in XML parsers. It is an another validation stage beyond parsing.

The parser reads a raw XML document, ensures that is well-formed, and may validate the document against a DTD or schema. Parser are of two types,

a. Validating Parser

b. Non-Validating Parser

a. A non validating parser checks if a document follows the XML syntax rules. It builds a tree structure from the tags used in XML document and return an error only when there is a problem with the syntax of the document. Non validating parsers process a document faster because they do not have to check every element against a DTD. In other words, these parsers check whether an XML document adheres to the rules of well formed document. The Expat parser is an example of non validating parser.

b. A Validating parser checks the syntax, builds the tree structure, and compares the structure of the XML document with the structure specified in the DTD associated with the document. In other words, in addition to checking whether an XML document is well formed, validating parsers also check whether the XML document adheres to the rules in the DTD used by the XML document. Microsoft MSXML parser is an example of a validating parser.

There are two standard APIs for parsing XML documents :

a. SAX (Simple API for XML)

b. DOM (Document Object Model)

a. SAX is an event-driven API. It defines a number of callback methods, which will be called when events occur during parsing. The SAX parser reads an XML document and generates events as it finds elements, attributes, or data in the document. There are events for document start, document end, attributes, text context, element start-tags, and element end-tags, entities, processing instructions, comments and others.

Advantages of SAX :

1. It is simple and memory efficient.
2. It is very fast and works for huge documents.

Disadvantages of SAX :

1. It is event-based so its API is less intuitive.
2. Clients never know the full information because the data is broken into pieces.

b. DOM is an object-oriented API. It explicitly builds an object model, in the form of a tree structure, to represent an XML document. The application can manipulate the nodes in the tree. DOM is a platform- and language-independent interface for processing XML documents. The

DOM API defines the mechanism for querying, traversing and manipulating the object model built.

Advantages of DOM :

1. It supports both read and write operations and the API is very simple to use.
2. It is preferred when random access to widely separated parts of a document is required.

Disadvantages of DOM :

1. It is memory inefficient. (Consumes more memory because the whole XML document needs to loaded into memory).
2. It is comparatively slower than other parsers.

❑ **Check Your Progress – 3 :**

1. Kind of Parsers are

   a. well-formed b. validating

   c. non-validating d. Both b & c

7.6 Document Type Definition (DTD) :

Similar to HTML, XML with correct syntax is Well Formed XML. That is a well formed XML document is a document that conforms to the XML syntax rules that were described in the previous sections. More specifically, to be well formed, an XML document must be validated against a Document Type Definition (DTD). The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. A DTD can be specified internally or externally.

- **Internal DTD**

A DTD is referred to as an internal DTD if elements are declared within the XML files. To refer it as internal DTD, standalone attribute in XML declaration must be set to yes. This means, the declaration works independent of external source.

The following is an example of internal DTD for the communication:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE communication [
<!ELEMENT communication (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<communication>
<to>Saurav</to>
<from>vinod</from>
<heading>congratulation</heading>
<body>Becoming BCCI President!</body>
</communication>
```

The above DTD can be interpreted like this :

!DOCTYPE communication (in line 2) defines that this is a document of the type communication. !ELEMENT communication (in line 3) defines the communication element as having four elements: "to,from,heading,body". !ELEMENT to (in line 4) defines the to element to be of the type "#PCDATA". !ELEMENT from (in line 5) defines the from element to be of the type "#PCDATA" and so on ...

- **External DTD**

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as no. This means, declaration includes information from the external source.

If the DTD is external to your XML source file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE communication SYSTEM "communication.dtd">
<communication>
<to>Saurav</to>
<from>vinod</from>
<heading>congratulation</heading>
<body>Becoming BCCI President!</body>
</communication>
```

And the following is a copy of the file "communication.dtd" containing the DTD :

```
<!ELEMENT communication (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

W3C supports an alternative to DTD called XML Schema. The W3C XML specification states that a program should not continue to process an XML document if it finds a validation error. The reason is that XML software should be easy to write, and that all XML documents should be compatible.

With HTML it was possible to create documents with lots of errors (like when you forget an end tag). One of the main reasons that HTML browsers are so big and incompatible, is that they have their own ways to figure out what a document should look like when they encounter an HTML error. With XML this should not be possible.

- **Element declarations in DTD**

One element declaration for each element type:

<!ELEMENT element_name content_specification>

where content_specification can be,

- (#PCDATA) parsed character data
- (child) one child element
- (c1,…,cn) a sequence of child elements c1…cn
- (c1|…|cn) one of the elements c1…cn

For each component c, possible counts can be specified as:

- c exactly one such element
- c+ one or more
- c* zero or more
- c? zero or one

Plus arbitrary combinations using parenthesis can be written as following:

<!ELEMENT f ((a|b)*,c+,(d|e))*>

- Elements with mixed content can be written as:

  <!ELEMENT text (#PCDATA | index | cite | glossary)*>

- Elements with empty content can be written as:

  <!ELEMENT image EMPTY>

- Elements with arbitrary content can be written as:

  <!ELEMENT data ANY>

- Attribute declarations in DTD
- Attributes are declared per element:

  <!ATTLIST section number CDATA #REQUIRED
  title CDATA #REQUIRED>

The above code declares two required attributes for element section.

The Possible attribute defaults are :

- #REQUIRED is required in each element instance
- #IMPLIED is optional
- #FIXED default always has this default value
- default has this default value if the attribute is omitted from the element instance
- **Attribute types in DTDs are,**
- CDATA represents string data
- (A1|…|An) represents enumeration of all possible values of the attribute (each is XML name)
- ID represents unique XML name to identify the element
- IDREF refers to ID attribute of some other element (intra-document link)
- IDREFS represents list of IDREF, separated by white space

❑ **Check Your Progress – 4 :**

1. DTD includes the specifications about the markup that can be used within the document, the specifications consists of all EXCEPT

   a. the browser name b. the size of element name

   c. entity declarations d. element declarations

7.7 XML Schema :

An XML Schema describes the structure of an XML document, just like a DTD. An XML document with correct syntax is called "Well Formed". An XML document validated against an XML Schema is both "Well Formed" and "Valid". XML Schema is an XML-based alternative to DTD.

- **XML Schema**

```
<xs:element name="communication">
<xs:complexType>
<xs:sequence>
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
<xs:element name="heading" type="xs:string"/>
<xs:element name="body" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

The Schema above is interpreted like this :

- <xs:element name="communication"> defines the element called "communication"
- <xs:complexType> the "communication" element is a complex type
- <xs:sequence> the complex type is a sequence of elements
- <xs:element name="to" type="xs:string"> the element "to" is of type string (text)
- <xs:element name="from" type="xs:string"> the element "from" is of type string
- <xs:element name="heading" type="xs:string"> the element "heading" is of type string
- <xs:element name="body" type="xs:string"> the element "body" is of type string

XML Schemas are written in XML. They are extensible to additions. They support data types and namespaces. With XML Schema, your XML files can carry a description of its own format. With XML Schema, independent groups of people can agree on a standard for interchanging data. With XML Schema, you can verify data.

- **XML schemas support Data Types**

One of the greatest strength of XML Schemas is the support for data types :

- It is easier to describe document content
- It is easier to define restrictions on data
- It is easier to validate the correctness of data
- It is easier to convert data between different data types

- **XML schemas use XML syntax**

Another great strength about XML Schemas is that they are written in XML:

- You don't have to learn a new language
- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schemas with the XML DOM
- You can transform your Schemas with XSLT

❑ **Check Your Progress – 5 :**

1. A schema describes

(i) grammer (ii) vocabulary

(iii) structure (iv) datatype of XML document

a. (i) & (ii) are correct b. (i), (iii), (iv) are correct

c. (i), (ii), (iv) are correct d. (i), (ii), (iii), (iv) are correct

7.8 Let Us Sum Up :

XML is now as important for the Web as HTML was to the foundation of the Web. XML is everywhere. It is the most common tool for data transmissions between all sorts of applications, and is becoming more and more popular in the area of storing and describing information. XML is now a technology that is part of every modern web browser which also include a stylesheet interpreter; XML files can now be sent directly without having to be translated into HTML beforehand. XML is the encoding for upper level languages such as RDF for defining information about documents and for OWL to define ontologies.

7.9 Answer for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (b)

❑ **Check Your Progress 2 :**

1. (c)

❑ **Check Your Progress 3 :**

1. (d)

❑ **Check Your Progress 4 :**

1. (a)

❑ **Check Your Progress 5 :**

1. (d)

7.10 Glossary :

1. **XML declaration :** The processing instruction that identifies a document as an XML document and contains the version attribute and the optional standalone and encoding attributes. An XML declaration is the first line in an XML document.
2. **Well-formed XML :** An XML document is well-formed if there is one root element, and all its child elements are nested within each other.

Start tags must have end tags, and each empty tag must be designated as such with a trailing slash (<emptyTag/>). Also all attributes must be quoted, and all entities must be declared.

3. **Valid XML :** XML that meets the constraints defined by its Document Type Declaration.
4. **Schema :** A technology-neutral term for the definition of the structure of an XML document.
5. **XSLT :** Extensible Stylesheet Language Transformations. An XML application that defines how an XML document will be transformed from one form of XML to another. XSLT is commonly used to transform XML data to HTML for rendering on a client.

7.11 Assignment :

1. Define XML. Discuss various advantages and disadvantages of XML.

7.12 Further Readings :

1. https://www.w3schools.com/xml/

BLOCK SUMMARY :

In this block, students have learnt and understand about Java Networking concept related to connection among two or more computing devices. The block gives an idea on the study and concept of class library in JDK along with querying of database. The students have been well explained on the concepts of Java Database Connectivity and ResultSet object.

The block detailed about the basic of Prepared Statement sub-interface using statement techniques. The concept related User Datagram Protocol along with JDBC Statement, CallableStatement and PreparedStatement are well explained to the students. The student will be demonstrated practically about JDBC network driver. The last unit also explored the use of XML in modern application for the proper and efficient manipulation of information and data.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Differentiate Stored procedure and function.
2. Explain the function of ResultSet object ?
3. Write note on JDBC DriverManager ?
4. Write short note on Socket and ServerSocket ?
5. What is XML? State its advantages.

❖ **Long Questions :**

1. Write short notes on User Datagram Protocol ?
2. Write short note on Querying the database ?
3. Write note on Java Networking ?
4. Discuss different steps to Connect with Database in JDBC.
5. Write a JDBC program to navigate records of salesman.
6. Write a XML file to demonstrate the use of DTD and Schema.

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	5	6	7
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-304

OBJECT ORIENTED CONCEPTS AND PROGRAMMING – II (ADVANCE JAVA)

BLOCK 3 : RMI & JAVABEANS

RMI & JAVABEANS

Block Introduction :

The RMI architecture is based on one important principle: the definition of behaviour and the implementation of that behaviour are separate concepts. In an RMI application, if we restart RMI registry, it will clean and re-compile all Java classes which is normally not recompiled completely.

In this block, we will detail about the basic steps of creating RMI Application along with steps of designing application on same system. The block will focus on the study and concept of Beans Development Kit, Bound Properties and Indexed Properties. The students will give an idea on working of RMI application.

In this block, the student will made to learn and understand about the concept related to JNDI Programming techniques. The concept related to Java Beans Concepts and Programming of JNDI is explained to students. The student will be demonstrated practically about exploring Javax.naming package technique.

Block Objectives :

After learning this block, you will be able to understand:

- Understanding about RMI Architecture
- Features of working of RMI
- Detailed steps involved in creating RMI Application
- Concept of Beans Development Kit
- Explain about writing a Simple Bean
- Characteristics about various Properties
- Concept of JNDI programming

Block Structure :

Unit
08

REMOTE METHOD INVOCATION

UNIT STRUCTURE

8.0 Learning Objectives :

After learning this Unit, you will be able to understand:

- RMI Architecture
- How RMI works
- Steps to create an RMI Application

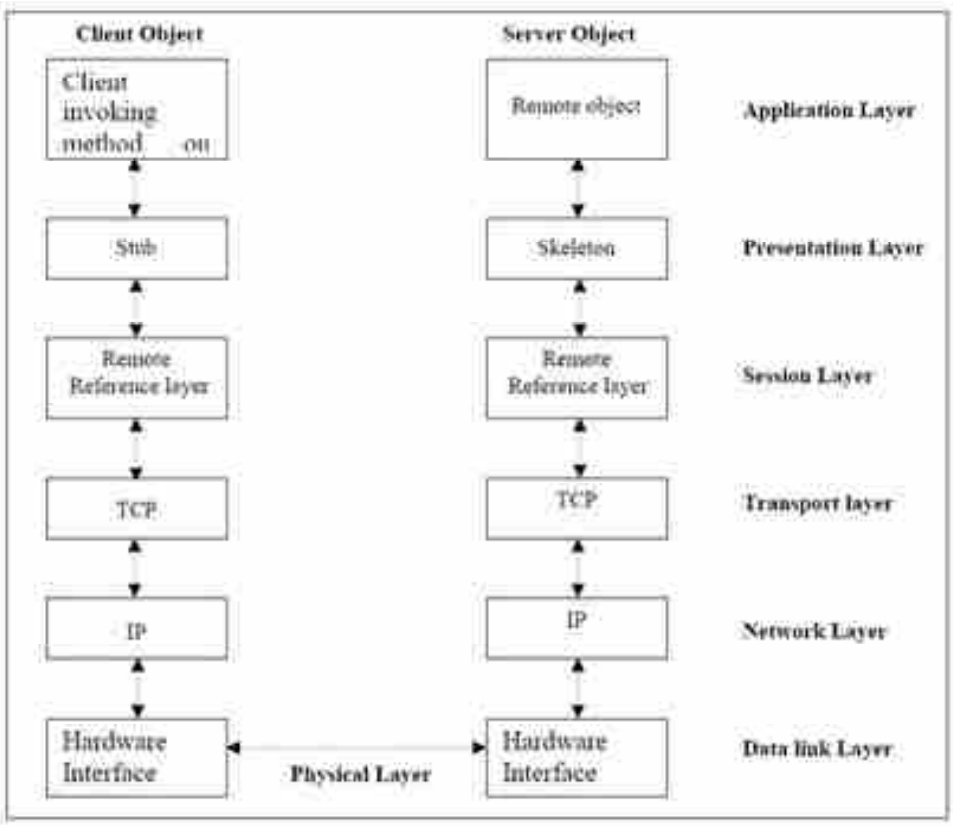
8.1 Introduction :

The RMI which is called as Remote Method Invocation is an API which gives a mechanism to create distributed application in java. It allows an object to call upon methods on an object running in another JVM. This gives remote communication that occurs among applications using two objects stub and skeleton.

It is way by which a programmer with the help of Java programming language and development environment will able to write object-oriented programming where objects on different computers gets interacted in distributed network. It is the Java version of remote procedure call, having an ability to pass one or several objects along the request.

8.2 RMI Architecture :

The RMI architecture is based on one important principle: the definition of behaviour and the implementation of that behaviour are separate concepts. RMI allows the code that defines the behaviour and the code that implements the behaviour to remain separate and to run on separate JVMs.



RMI Architecture

Specifically, in RMI, the definition of a remote service is coded using a Java interface. The implementation of the remote service is coded in a class. Therefore, the key to understanding RMI is to remember that interfaces define behaviour and classes define implementation. Fig 1.2 shows separation :

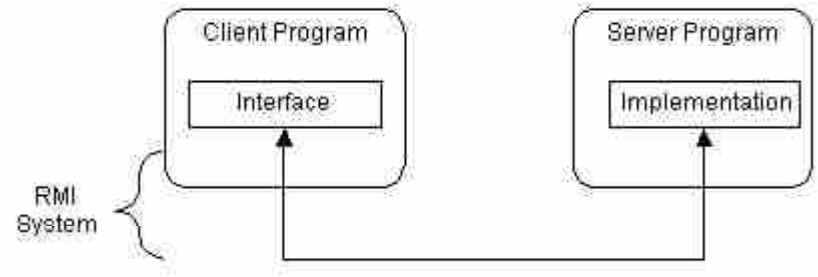


Fig 1.2 Separation of program

It is noted that a Java interface will not have executable code. RMI supports two classes that implement the same interface:

First class is implementation of behaviour that runs on server

Second class acts as proxy for remote service which runs on client

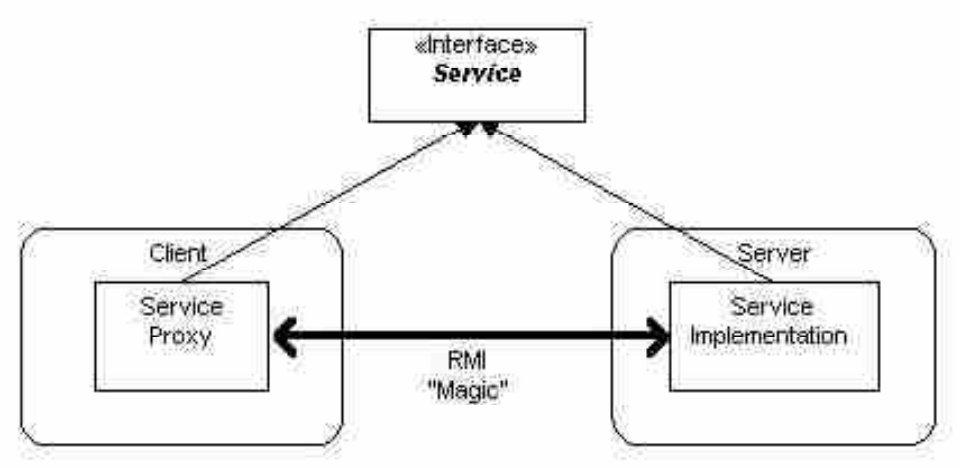


Fig 1.3 Proxy remote service

A client program makes method calls on the proxy object, RMI sends the request to the remote JVM, and forwards it to the implementation. Any return values provided by the implementation are sent back to the proxy and then to the client's program.

In a high-level RMI architecture, there are three abstraction layers :

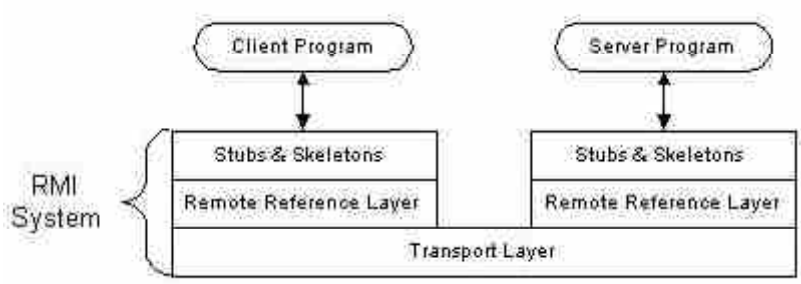


Fig 1.4 RMI layers

First layer is Stub and Skeleton layer that is located just below view of developer which intercepts method calls created by client to interface reference variable by redirecting calls to a remote RMI service.

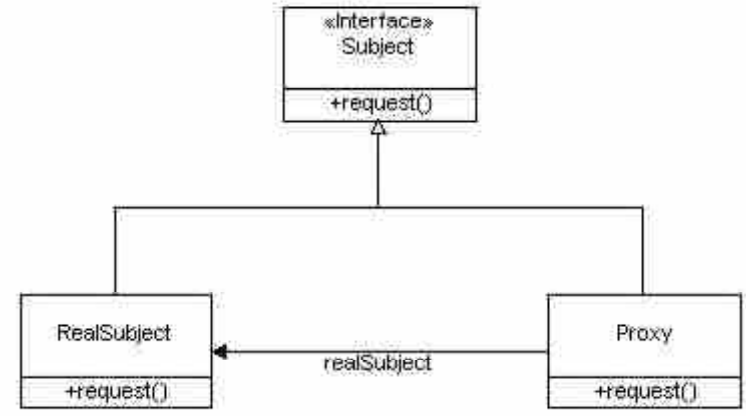


Fig 1.5 Stub and Skeleton layer

Second layer is Remote Reference Layer which show how to interpret and manage references made from clients to remote service objects. It is found that such layer will connect clients to remote service objects which are running and exported on server.

Third layer is transport layer which is based on TCP/IP connections that exist among machines in a network showing connectivity and firewall penetration strategies.

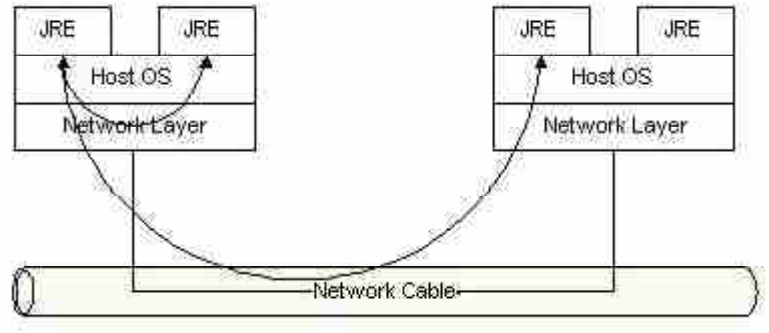


Fig 1.6 Transport layer

❑ **Check Your Progress – 1 :**

1. RMI includes _____.

   a. Interface which defines behavior

   b. Class which defines implementation

   c. Both of these

   d. None of these

8.3 How RMI Works :

In RMI we see that the code from machine A gets accessed for object which is residing on machine B in remote relocation. In this, there appears two intermediate objects which is known as "stub? and "skeleton? that really handle the communication. We see that these two objects performs following tasks :

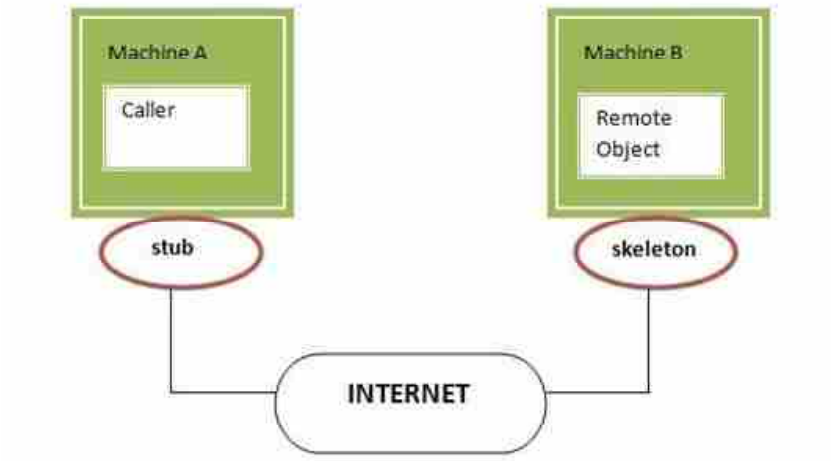


Fig 1.7 Task performed by stub and skeleton

Task which are handled by stub object on machine A :

A. Building a information block which consists of

- An identifier of the remote object
- An operation number that describes the method to be called
- The method parameters called marshaled parameters that are to be encoded into a suitable format for transporting over the network

Tasks which are handled by skeleton object on machine B :

B. Sending the information to the server.

- Unmarshalling the parameters
- Calling upon required method of the object which is lying on the server
- Capturing and returning the value if successful or an exception if unsuccessful , after the call on the server
- Marshalling the returned value
- Packaging the value that is in the marshalled form and sending to the stub on the client.

Subsequently the stub object unmarshals the return value or exception from the server as the case may be. This is the returned value of the remote method invocation. If an exception is thrown, the stub object will throw again the exception to the caller.

❑ **Check Your Progress – 2 :**

1. If two machines A and B wants to continue the communication, then which of the following intermediate object will act on Machine A

   a. Stub b. Skeleton

   c. Both of these d. None of these

8.4 Steps to Create an RMI Application :

An RMI application carries a Client program and Server program. We see that a Server program will form many remote object which make their references available for client to call upon method on it. It is seen that a Client program will make request for remote objects on server and will call upon method on them. It is found that Stub and Skeleton are two important object which are applied for communication with remote object.

Following are the steps required to create an RMI application :

- Defining remote interface.
- Implementing remote interface.
- Creating and starting remote application
- Creating and starting client application

Define a remote interface

We see that remote interface will specify methods that can be called upon remotely by a client. In such case clients program will communicate to remote interfaces and not to classes to implement it. To be a remote interface, a interface must extend the Remote interface of java.rmi package.

```
import java.rmi.*;
public interface AddServerInterface extends Remote
{
public int sum(int a,int b);
}
```

Implementation of remote interface

To implement remote interface, a class should either extend UnicastRemoteObject or use exportObject() method of UnicastRemoteObject class.

```
import java.rmi.*;
import java.rmi.server.*;
public clss Adder extends UnicastRemoteObject implements AddServerInterface
{
Adder()throws RemoteException{
super();
}
public int sum(int a,int b)
{
return a+b;
}
}
```

Create AddServer and host rmi service

We see that initially, we need to create server application and host rmi service Adder in it. It can be done using rebind() method of java.rmi.Naming class. It is noted that rebind() method will carry two arguments where first shows name of object reference while second argument is reference to instance of Adder

```
import java.rmi.*;
import java.rmi.registry.*;
public class AddServer{
public static void main(String args[]){
try{
AddServerInterface addService=new Adder();
Naming.rebind("AddService", addService);
//addService object is hosted with name AddService.
}catch(Exception e) {System.out.println(e);}
}
}
```

Create client application

It is found that client application carry java program which calls upon the lookup() method of the Naming class. It accepts one argument, rmi URL and returns reference to object of type AddServerInterface. In this, every remote method invocation is done on this object.

```
import java.rmi.*;
public class Client{
public static void main(String args[]){
```

```
try{
AddServerInterface
st=(AddServerInterface)Naming.lookup("rmi://}+args[0]+"/AddService");
System.out.println(st.sum(25,8));
}catch(Exception e){System.out.println(e);}
}
}
```

❑ **Check Your Progress – 3 :**

1. Which of the following steps are required to create RMI application ?
   a. Defining and implementing remote interface.
   b. Creating and starting remote application
   c. Creating and starting client application
   b. All of these

8.5 Steps for Deploying RMI Application (on same system) :

There are certain steps applied in deploying RMI application on similar systems :

Step 1 : Create interface and compile it using javac ArithInt.java

Step 2 : Create an implementation and compile with javac ArithImpl.java

Step 3 : Create server program which registers implementation and compile it using javac ArithServer.java

Step 4 : Create stub and skeleton using rmi compiler which is rmic ArithImpl

Step 5 : Create and compile client application using javac ArithClient.java

Step 6 : Set classpath in directory type by setting classpath=% classpath%;.;

Step 7 : Start RMI registry by opening new command window and typing rmiregistry

Step 8 : Run server application by opening new command window and setting classpath type java ArithServer

Step 9 : Run client application by opening new command window and further setting classpath type java ArithClient.

❑ **Check Your Progress – 4 :**

1. RMI stands for _____.
   a. Resource Method Interaction b. Remote Method Invocation
   c. Reverse Media Invocation d. None of these

8.6 Troubleshooting :

In an RMI application, if we restart RMI registry, it will clean and re-compile all Java classes which is normally not recompiled completely. If you're trying to start RMI server, and it complains about "Class not found" for class which is already present, then the RMI registry gets corrupted and in such scenario we need to restart.

For restarting, we have to make sure that RMI registry is being started with empty classpath, and RMI servers gets started using valid java.rmi.server.codebase. It is found that normally, it is possible to start RMI registry having real classpath, which makes many problems to go away and makes too complicated in long run as it covers many problems with codebase property. It is noted that for codebase property to point to a file in URL if you are not having client download copies of stub files. It seems that the java.policy file gets effected when RMI registry starts that will take care of all RMI classes attached to the registry.

❑ **Check Your Progress – 5 :**

1. What is RMI registry ?

   a. It is an object that is provided on another computer to request the object

   b. It is an object of the server side that receive the request and respond to the request of the client

   c. It is the registry that keeps the look ups of all the objects in a list by given name.

   d. None of these

8.7 Advanced RMI Concepts :

Using RMI infrastructure and packages, RMI based Java clients remotely calls upon certain methods on RMI based Java server objects. Such type of RMI based server objects runs in separate JVM?s on same machine or on another machine host.

The theoretical prototype of RMI is Remote Procedure Calls which is inside object oriented evolution of technology. The RMI is abstractly comparable to other distributed object architectures such as CORBA and DCOM. The RMI and CORBA paradigms are integrating using support for IIOP - IIOP which is Internet Inter-Orb Protocol.

It is seen that RMI clients will communicate transparently with RMI server objects by calling upon methods on a client side proxy object which serializes the method parameters as well as streams them to required server object instance where return parameters gets unmarshalled in similar manner. RMI allows for clients and servers to pass objects as method parameters and return values. It is noted that the difference among RMI and CORBA is RMI?s ability to pass objects by value.

❑ **Check Your Progress – 6 :**

1. CORBA stands for _____.

   a. Common Object Request Broker Architecture

   b. Computing Object resources Business Architecture

   c. Common Object recourse Broker Architecture

   d. None of these

8.8 Let Us Sum Up :

In this unit we have learnt that RMI is Remote Method Invocation which is an API showing mechanism to create distributed application in java by allowing object to call upon methods on an object running in another JVM.

It is seen that RMI architecture depends on important principle as behaviour and the implementation of particular behaviour having different concepts.

It is found that RMI application has Client program and Server program where Server program form many remote object making references available for client to call upon method on it.

With RMI infrastructure and packages, RMI based Java clients remotely calls upon certain methods on RMI based Java server objects.

8.9 Answer for Check Your Progress :

- **Check Your Progress 1 :**

  1. (c)

- **Check Your Progress 2 :**

  1. (a)

- **Check Your Progress 3 :**

  1. (d)

- **Check Your Progress 4 :**

  1. (b)

- **Check Your Progress 5 :**

  1. (c)

- **Check Your Progress 6 :**

  1. (a)

8.10 Glossary :

1. **RMI :** It is Remote Method Invocation an API which create distributed application in java.
2. **RMI Architecture :** It is an arrangement that depends on behaviour and implementation concepts.
3. **RMI Application :** It is a Client and Server program

8.11 Assignment :

Explain the working of RMI ?

8.12 Activities :

Study about Stub and Skeleton layers.

8.13 Case Study :

Study the steps involved in create an RMI Application.

8.14 Further Readings :

1. Operating System Concept by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

Unit 09

JAVABEANS – 1

UNIT STRUCTURE

9.0 Learning Objectives :

After learning this Unit, you will be able to:

- The Beans Development Kit
- Writing a Simple Bean

9.1 Introduction :

With respect to Java white paper that can be reusable software component. A bean sums up various objects into single object in order to access object from various places. Moreover, it provides the easy maintenance. It is developed by Sun Microsystems that shows how Java objects interact and conforms to certain specification that is identical to ActiveX control. It is applied by several application which understands JavaBeans format. It is noted that the main difference among ActiveX controls and JavaBeans are that ActiveX controls can be developed in any programming language but can be worked out only on Windows platform, while JavaBeans can be developed only in Java and can run on any platform.

9.2 Java Beans Concepts :

JavaBeans is an object-oriented programming interface from Sun Microsystems that lets you build re-useable applications or program building blocks called components that can be deployed in a network on any major operating system platform. Like Java applets, JavaBeans components can be used to give World Wide Web pages interactive capabilities such as computing interest rates or varying page content based on user or browser characteristics.

From a user's point-of-view, a component can be a button that you interact with or a small calculating program that gets initiated when you press the button. From a developer's point-of-view, the button component and the calculator

component are created separately and can then be used together or in different combinations with other components in different applications or situations.

When the components or Beans are in use, the properties of a Bean are visible to other Beans and Beans that haven't "met" before can learn each other's properties dynamically and interact accordingly.

Beans are developed with a Beans Development Kit (BDK) from Sun and can be run on any major operating system platform inside a number of application environments including browsers, word processors, and other applications.network.

Beans also have persistence, which is a mechanism for storing the state of a component in a safe place. This would allow, for example, a component to "remember" data that a particular user had already entered in an earlier user session.

The JavaBeans API makes it possible to write component software in the Java programming language. Components are self-contained, reusable software units that can be visually composed into composite components, applets, applications, and servlets using visual application builder tools. JavaBean components are known as Beans.

Components expose their features to builder tools for visual manipulation. A Bean's features are exposed because feature names adhere to specific design patterns. A "JavaBeans-enabled" builder tool can then examines the Bean's patterns, discern its features, and expose those features for visual manipulation. A builder tool maintains Beans in a palette or toolbox. You can select a Bean from the toolbox, drop it into a form, modify it's appearance and behaviour, define its interaction with other Beans, and compose it and other Beans into an applet, application, or new Bean. All this can be done without writing a line of code.

Features :

- Support for introspection allowing a builder tool to analyze how a bean works.
- Support for customization allowing a user to alter the appearance and behaviour of a bean.
- Support for events allowing beans to fire events, and informing builder tools about both the events they can fire and the events they can handle.
- Support for properties allowing beans to be manipulated programmatically, as well as to support the customization mentioned above.

Support for persistence allowing beans that have been customized in an application builder to have their state saved and restored. Typically persistence is used with an application builder's save and load menu.

To build a component with JavaBeans, you write language statements using Sun's Java programming language and include JavaBeans statements that describe component properties such as user interface characteristics and events that trigger a bean to communicate with other beans in the same container or elsewhere in the commands to restore any work that has gone into constructing an application.

❑ **Check Your Progress – 1 :**

1. What is Java Bean ?

   a. It is a class

   b. It is a Interface

   c. It is a software component that has been designed to be reusable in a variety of different environments

   d. All of these

9.3 The Beans Development Kit :

The Beans Development Kit (BDK) is intended to support the early development of JavaBeansTM components and to act as a standard reference base for both bean developers and tool vendors. The BDK provides a reference bean container, the "BeanBox" and a variety of reusable example source code (in the demo and beanbox subdirectories) for use by both bean developers and tools developers.

The BDK is not intended for use by application developers, nor is it intended to be a full-fledged application development environment. Instead application developers should check the various Java application development environments supporting JavaBeans.

We see that the general description of Beans Development Kit files and directories are shown below:

- README.html contains an entry point to the BDK documentation
- LICENSE.html contains the BDK license agreement
- GNUmakefile and Makefile are Unix and Windows makefiles (.gmk and .mk suffixes) for building the demos and the BeanBox, and for running the BeanBox
- beans/apis contains java directory having JavaBeans source files and sun directory having property editor source files
- beans/beanbox contains makefiles for creating BeanBox with scripts for running BeanBox. It carries classes directory having BeanBox class files along with lib directory having BeanBox support jar file used by MakeApplet's produced code. In this the sun and sunw directories has BeanBox source (.java) files with tmp directory that will directly generate event adapter source and class files, .ser files, and applet files directly by MakeApplet
- beans/demos has makefiles for creating demo Beans. It has HTML directory that carry applet wrapper that run in appletviewer, HotJava, or JDK1.1-compliant browsers. It is further noticed that a sun directory has wrapper directory having Bean applet wrapper along with demos directory for demo source file.
- beans/doc has demos documentation along with javadoc directory having JavaBeans and JavaBeans-related class along with interface and miscellaneous documentations.
- beans/jars contains jar files for demo Beans

❑ **Check Your Progress – 2 :**

1. What is the use of README html in java beans ?

   a. It contains the BDK license agreement

   b. It contains an entry point to the BDK documentation

   c. It contains jar files

   d. All of these

9.4 Writing a Simple Bean :

Beans are designed by adding minimal amount of code to an existing class definition so as to turn class into JavaBean. In this, you need to add a pair of methods to existing class definition so as to make a Bean which can be applied either by modifying source for existing class, or by extending behaviour of existing class by sub classing it.

From the initial impression, we see that there appears no difference among JavaBean and normal Java AWT component object. There results minor differences, such as fact which Beans need not be visible components, Java you need only follow a few simple rules to make Beans out of existing classes. So writing simple bean involves following steps:

- Creating a simple bean
- Compiling the bean
- Generating a Java Archive (JAR) file
- Loading the bean into the GUI Builder of the NetBeans IDE
- Inspecting the bean's properties and events

Initially, the bean will be named as SimpleBean and we needs following steps to create it. Writing SimpleBean code and keeping it in file SimpleBean.java, in directory be framed using following code :

```
import java.awt.Color;
import java.beans.XMLDecoder;
import javax.swing.JLabel;
import java.io.Serializable;
public class SimpleBean extendsJLabel
                implements Serializable {
  public SimpleBean() {
    setText( "Hello world!" );
    setOpaque( true );
    setBackground( Color.RED );
    setForeground( Color.YELLOW );
    setVerticalAlignment( CENTER );
    setHorizontalAlignment( CENTER );
  }
}
```

We see that simple bean extends javax.swing.JLabel graphic component and takes it properties that makes SimpleBean a visual component. It implements java.io.Serializable interface which are used by bean to implement Serializable or Externalizable interface.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project default="build">
<dimame property="basedir" file="${ant.file}"/>
<property name="beanname" value="SimpleBean"/>

<property name="jarfile" value="${basedir}/$[beanname}.jar"/>
  <target name="build" depend="compile">
    <jar destfile=?${jarfile}" basedir="${basedir}" include="*.class">
      <manifest>
        <section name="${beanname}.class>
          <attribute name="Java-Bean" value="true"/>
        </section>
      </manifest>
    </jar>
  </target>

<target name="compile">
    <javac destdir="${basedir}">
      <src location="${basedir}"/>
    </java>
</target>
<target name="clean">
    <delete file="${jarfile}">
      <fileset dir="${basedir}" includes="*.class"/>
    </delete>
  </target>
</project>
```

Now you need to create a manifest, the JAR file, and class file SimpleBean.class and apply the Apache Ant tool to form such files. Apache Ant is Java-based build tool which allows to generate XML-based configurations files :

It is recommended to save an XML script in the build.xml file, because Ant recognizes this file name automatically.

Load the JAR file :

To load the JAR file, apply NetBeans IDE GUI Builder to load the jar file as follows :

- Sart NetBeans

- From the File menu select "New Project" to create a new application for your bean. You can use "Open Project" to add your bean to an existing application.
- Create a new application using the New Project Wizard.
- Select a newly created project in the List of Project, expand the Source Packages noe, and select the Default Package element.
- Click the right mouse button and select New|JFrameForm from the pop-up menu.
- Select the newly created Form node in the Project Tree. A blank form opens in the GUI Builder view of an Editor tab.
- Open the Palette Manager for Swing/AWT components by selecting Palette Manager in the Tools menu.
- In the Palette Manager window select the beans components in the Palette tree and press the "Add from JAR" button.
- Specify a location for your SimpleBean JAR file and follow the Add from JAR Wizard instructions.
- Select the Palette and Properties options from the Windows menu.

❑ **Check Your Progress – 3 :**

1. Which of the following are java beans API ?

   a. AppletInitializer b. BeanInfo

   c. ExceptionListener d. All of these

9.5 Let Us Sum Up :

While studying this unit, we have learnt that Java Bean is java class that which does not have arg constructor but are serializable and gives methods to set and obtain values of properties that can be called as getter and setter methods.

It is found that JavaBeans is object-oriented programming interface from Sun Microsystems which allows to build re-useable applications or program building blocks that can be deployed in network on any major operating system platform.

Beans are developed with Beans Development Kit from Sun and can be run on any major operating system platform inside a number of application environments including browsers, word processors, and other applications.

It is noted that Beans Development Kit intends to support early development of JavaBeans components which act as standard reference base for both bean developers and tool vendors.

Beans are designed by adding minimal amount of code to an existing class definition so as to turn class into JavaBean.

9.6 Answer for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (c)

❑ **Check Your Progress 2 :**

1. (b)

❑ **Check Your Progress 3 :**

1. (d)

9.7 Glossary :

1. Java Bean : An object-oriented programming interface started by Sun Microsystems which build re-useable applications or program building blocks.

9.8 Assignment :

Write short note on Java Bean.

9.9 Activities :

Collect some information on Beans Development Kit.

9.10 Case Study :

Generalised the basic java.beans package and discuss.

9.11 Further Readings :

1. Operating System Concept by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

Unit
10

JAVABEANS – 2

UNIT STRUCTURE

10.0 Learning Objectives :

After learning this Unit, you will be able to:

Properties of Java Beans like Simple properties, bound properties, indexed properties and constrained properties.

10.1 Introduction :

A Java Bean is a java class that which does not have arg constructor but are serializable and gives methods to set and obtain values of properties that can be called as getter and setter methods

10.2 Properties :

To define a property in a bean class, supply public getter and setter methods.

For example, the following methods define an int property called mouthWidth :

```
public class FaceBean {
   private int mMouthWidth=90;
   public int getMouthWidth() {
      return mMouthWidth;
   }
   public void set.MouthWidth(int mw) {
```

```
        mMouthWidth=mw;
    }
}
```

It is noted that builder tool NetBeans identifies method names and shows mouthWidth property in list. It identifies the type, int, and provides correct editor so property can be manipulated at design time.

10.2.1 Simple Properties :

The SimpleBean properties will appear in the Properties window. For example, you can change a background property by selecting another color. To preview your form, use the Preview Design button of the GUI Builder toolbar. To inspect events associated with the SimpleBean object, switch to the Events tab of the Properties window.

10.2.2 Bound Properties :

A bound property inform listeners when its value changes. In java.beans package, in a class, PropertyChangeSupport,takes care of work of bound properties which further will keep track of property listeners including convenience method which fires property change events to all registered listeners.

In an example, we see that how a mouthWidth property can be prepared with bound property using PropertyChangeSupport. The necessary additions for the bound property are shown in bold.

```
import java.beans.*;
public class FaceBean {
    private int mMouthWidth=90;
    private PropertyChangeSupport mPcs=
        new PropertyChangeSupport(this);
    public int getMouthWidth() {
        return mMouthWidth;
    }

    public void setMouthWidth(int mw) {
        int oldMouthWidth = mMouthWidth
        mMouthWidth=mw;
        mPcs.firPropertyChange("mouthWidth, oldMouthWidth, mw);
    }
public void
    addPropertyChangeListener(PropertyChangeListener listene) {
        mPcs.addPropertyChangeListener(listener);
    }
```

```
    public void
    removePropertyChangeListener(PropertyChangeListener listene) {
mPcs.rmovePropertyChangeListener(listener);
        }
}
```

Bound properties will work directly with other bean properties using builder tool

10.2.3 Indexed Properties :

An indexed property is an arrangement of arrays rather than single value where bean class shows method for getting and setting all array. Here is an example for an int[] property called testGrades

```
public int[] getTestGrades() {
  return mTestGrades;
}

public void setTestGrades(int[] tg) {
  mTestGrades = tg;
}
```

For indexed properties, the bean class also provides methods for getting and setting a specific element of the array.

```
public getTestGrades(int index) {
  return mTestGrades[index];
}
public void setTestGrades(int index, int grade) {
  mTestGrades[index] = grade;
}
```

10.2.4 Constrained Properties :

A constrained property is special bound property where the bean keeps track of set of veto listeners. In case of change of constrained property, listeners are consulted about change. Any one of listeners has chance to veto change, where property remains unchanged.

The veto listeners are separate from the property change listeners. Fortunately, the java.beans package includes a VetoableChangeSupport class that greatly simplifies constrained properties.

Changes to the mouthWidth example are shown in bold :

```
import java.beans.*;
public class FaceBean {
  private int mMouthWidth = 90;
  private PropertyChangeSupport mPcs=
    new PropertyChangeSupport(this);
```

```
    private VetoableChangeSupport mVcs=
      new VetoableChangeSupport(this);
    public int getMouthWidth() {
      return mMouthWidth;
  }
    public void
    setMouthWidth(int mw) throws PropertyVetoException {
      int oldMouthWidth = mMouthWidth;
      mVcs.fireVetoableChange("mouthWidth", oldMouthWidth, mw);
      mMouthWidth = mw,
      mPcs.firePropertyChange("mouthWidth", oldMouthWidth, mw);
    }
      publc void
      addPropertyChangeListener(PropertyChangeListener listener) {
        mPcs.addPropertyChangeListener(listener);
      }

      public void
      removePropertyChangeListener(PropertyChangeListener listener) {
        mPcs.removePropertyChangeListener(listener);
      }
      public void
      addVetoableChangeListener(VetoableChangeListener listener) {
        mVcs.addVetoableChangeListener(listener);
      }
      public void
      removeVetoableChangeListener(VetoableChangeListener listener) {
        mVcs.removeVetoableChangeListener(listener);
```

❑ **Check Your Progress – 4 :**

1. What is Bound Properties of Beans ?

   a. It is an arrangement of arrays rather than single value

   b. It keeps track of set of veto listeners

   c. It inform listeners when its value changes

   d. None of these

10.3 Let Us Sum Up :

While studying this unit, we have learnt A bound property inform listeners when its value changes. In java.beans package, in a class, PropertyChangeSupport,takes care of work of bound properties which further will keep track of property listeners including convenience method which fires property change events to all registered listeners.

10.4 Answer for Check Your Progress :

❑ **Check Your Progress 4 :**

1. (c)

10.5 Glossary :

1. Beans Development Kit : It is intended to support early development of JavaBeans components which acts as standard reference base.

10.6 Assignment :

Write different properties of Java Beans and explain them each.

10.7 Activities :

Collect different properties of Java Beans.

10.8 Case Study :

Generalised the basic java.beans properties and discuss.

10.9 Further Readings :

1. Operating System Concept by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

Unit
11

JAVA NAMING AND DIRECTORY INTERFACE API

UNIT STRUCTURE

11.0 Learning Objectives :

After learning this Unit, you will be able tounderstand:

- JNDI Overview
- Understanding the Concepts behind JNDI Programming

11.1 Introduction :

The Java Naming and Directory Interface (JNDI) provides naming and directory functionality to applications written in the Java programming language. It is designed to be independent of any specific naming or directory service implementation. Thus a variety of services--new, emerging, and already deployed ones--can be accessed in a common way.

The Java Naming and Directory Interface (JNDI) is an application programming interface (API) for accessing different kinds of naming and directory services. JNDI is not specific to a particular naming or directory service, it can be used to access many different kinds of systems including file systems; distributed objects systems like CORBA, Java RMI, and EJB; and directory services like LDAP, Novell NetWare, and NIS+.

JNDI is similar to JDBC in that they are both Object-Oriented Java APIs that provide a common abstraction for accessing services from different vendors. While JDBC can be used to access a variety of relational databases, JNDI can be used to access a variety of of naming and directory services. Using one API to access many different brands of a service is possible because both

JDBC and JNDI subscribe to the same architectural tenet: Define a common abstraction that most vendors can implement. The common abstraction is the API. It provides an objectified view of a service while hiding the details specific to any brand of service. The implementation is provided by the vendor, it plugs into the API and implements code specific to accessing that vendor's product.

11.2 Naming and Directory Service :

A fundamental facility in any computing system is the naming service--the means by which names are associated with objects and objects are found based on their names. When using almost any computer program or system, you are always naming one object or another. For example, when you use an electronic mail system, you must provide the name of the recipient. To access a file in the computer, you must supply its name. A naming service allows you to look up an object given its name.

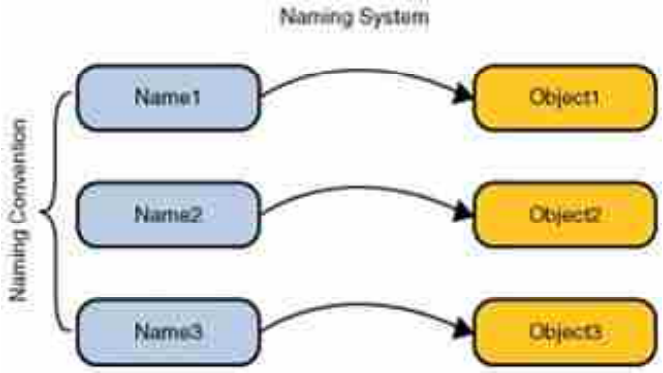


Fig 3.1 Naming Service

To look up an object in a naming system, you supply it the name of the object. The naming system determines the syntax that the name must follow. This syntax is sometimes called the naming systems naming convention. A name is made up components. A name's representation consist of a component separator marking the components of the name.

Directory :

Many naming services are extended with a directory service. A directory service associates names with objects and also associates such objects with attributes.

directory service = naming service + objects containing attributes

You not only can look up an object by its name but also get the object's attributes or search for the object based on its attributes.

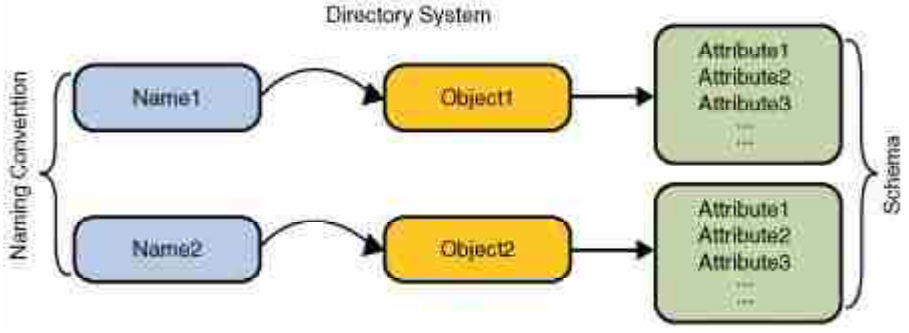


Fig 3.2 Directory service

A directory object represents an object in a computing environment. A directory object can be used, for example, to represent a printer, a person, a computer, or a network. A directory object contains attributes that describe the object that it represents. A directory object can have attributes. For example, a printer might be represented by a directory object that has as attributes its speed, resolution, and color. A user might be represented by a directory object that has as attributes the user's e-mail address, various telephone numbers, postal mail address, and computer account information.

A directory is a connected set of directory objects. A directory service is a service that provides operations for creating, adding, removing, and modifying the attributes associated with objects in a directory. The service is accessed through its own interface.

❑ **Check Your Progress – 1 :**

1. What is the purpose of directory object ?

   a. It represents an object in a computing environment.

   b. It is just a simple folder to hold files

   c. Both of these

   d. None of these

11.3 Enter JNDI :

The Java Naming and Directory InterfaceTM (JNDI) provides naming and directory functionality to applications written in the JavaTM programming language. It is designed to be independent of any specific naming or directory service implementation. Thus a variety of services--new, emerging, and already deployed ones--can be accessed in a common way.

The JNDI architecture consists of an API (Application Programming Interface) and an SPI (Service Provider Interface). Java applications use this API to access a variety of naming and directory services. The SPI enables a variety of naming and directory services to be plugged in transparently, allowing the Java application using the API of the JNDI technology to access their services.

Specifying a Resource Reference

- The instructions that follow mention the AccountEJB example, which is described in the section, A Bean-Managed Persistence Example. The AccountEJB code is in the examples/src/ejb/account directory.
- In the deploytool, select the component from the tree.
- Select the Resource Ref's tab.
- Click Add.
- In the Coded Name field, enter jdbc/AccountDB.
- The AccountEJB refers to database private String dbName = "java:comp/env/jdbc/AccountDB";
- The java:comp/env prefix is the JNDI subcontext for the component. Becuase this subcontext is implicit in the Coded Name field, you don't need to include it there.

- In the Type combo box, select javax.sql.DataSource. A DataSource object is a factory for database connections.
- In the Authentication combo box, select Container.

Mapping a Resource Reference to a JNDI Name

- Select the J2EE application from the tree.
- Select the JNDI Names tab.
- In the References table, select the row containing the resource reference. For the AccountEJB example, the resource reference is jdbc/AccountDB, the name you entered in the Coded Name field of the Resource Ref's tab.
- In the row you just selected, enter the JNDI name. For the AccountEJB example, you would enter jdbc/Cloudscape in the JNDI Name field.

❑ **Check Your Progress – 2 :**

1. What is JNDI ?
   a. It provides naming and directory functionality to applications written in the java
   b. It is independent of any specific naming or directory service implementation.
   c. It helps in accessing variety of services like new, emerging technology.
   d. All of these

11.4 JNDI Overview :

The Java Naming and Directory Interface (JNDI) is part of the Java platform, providing applications based on Java technology with a unified interface to multiple naming and directory services. You can build powerful and portable directory-enabled applications using this industry standard.

JNDI is an API specified in Java technology that provides naming and directory functionality to applications written in the Java programming language. It is designed especially for the Java platform using Java's object model. Using JNDI, applications based on Java technology can store and retrieve named Java objects of any type. In addition, JNDI provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes.

JNDI is also defined independent of any specific naming or directory service implementation. It enables applications to access different, possibly multiple, naming and directory services using a common API. Different naming and directory service providers can be plugged in seamlessly behind this common API. This enables Java technology-based applications to take advantage of information in a variety of existing naming and directory services, such as LDAP, NDS, DNS, and NIS(YP), as well as enabling the applications to coexist with legacy software and systems.

❑ **Check Your Progress – 3 :**

1. What is LDAP ?

   a. Lightweight Directory Access Protocol

   b. Light Directory Access Program

   c. Lightweight Directory Access Program

   d. None of these

11.5 Understanding the Concepts behind JNDI Programming :

Java Naming and Directory Interface allows distributed applications to look up services in an abstract, resource-independent way. The most common use case is to set up a database connection pool on a Java EE application server. Any application that's deployed on that server can gain access to the connections they need using the JNDI name java:comp/env/FooBarPool without having to know the details about the connection. This has several advantages:

- If you have a deployment sequence where apps move from devl->int->test->prod environments, you can use the same JNDI name in each environment and hide the actual database being used. Applications don't have to change as they migrate between environments.
- You can minimize the number of folks who need to know the credentials for accessing a production database. Only the Java EE app server needs to know if you use JNDI.

The Java Naming and Directory Interface is an application programming interface (API) having naming and directory functions to applications written using the Java programming language. It is defined to be independent of any specific directory service implementation. Thus a variety of directories--new, emerging, and already deployed--can be accessed in a common way.

❑ **Check Your Progress – 4 :**

1. Java Naming and Directory Interface is :

   a. an API

   b. uses Java programming language

   c. independent of any specific directory

   d. all of above

11.6 Programming with JNDI :

NDI provides an interface that supports all this common functionality. JNDI naming revolves around a small set of classes and a handful of operations. JNDI performs all naming operations relative to a context. To assist in finding a place to start, the JNDI specification defines an InitialContext class. This class is instantiated with properties that define the type of naming service in use and, for naming services that provide security, the ID and password to use when connecting.

The JNDI architecture consists of an API and a service provider interface (SPI). All Java applications that utilize naming and directory services use the JNDI API to access a variety of naming and directory services. The SPI on the other hand enables a variety of naming and directory services to be plugged in transparently, thereby allowing the Java application using the JNDI API to access their services. The JNDI API includes:

- Java 2 SDK
- v1.3

There are steps which configure and run JNDI Connection pool for an application :

(1) Configure data-source in Server and create JNDI name.

(2) Configure web.xml

(3) Configure Spring bean with JNDI Datasource

(4) Include JDBC driver library on Server lib

JNDI style lookup is used consistently throughout EJB applications. Client programs have symbolic names for the server objects that they wish to use - these are looked up at run-time to obtain initial references for those objects. Session and Entity beans that use databases again have symbolic names for these databases; the actual database details being defined at deployment time when the appropriate data are inserted into the tables of a JNDI service associated with the EJB system.

The JNDI subsystem is used in an Application Server as a directory for such objects as resource managers and Enterprise JavaBeans (EJBs). Objects managed by the WebLogic container have default environments for getting the JNDI InitialContext loaded when they use the default InitialContext() constructor. For a Collaboration using a WebLogic EJB Object Type Definition (OTD) to find the home interface of an EJB, the JNDI properties must be configured and associated with the OTD. However, for other external clients, accessing the WebLogic naming service requires a Java client program that sets up the appropriate JNDI environment when creating the JNDI Initial Context. The example shows initial context to WebLogic JNDI from a stand-alone client :

```
HashMap env = newHashMap();
env.put (Context.PROVIDER_URL, "t3://localhost:7001/");
env.put (Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.WLInitialContextFactory");
Context initContext = new InitialContext (env);
```

Once an initial context is created, sub-contexts can be created, objects can be bound, and objects can be retrieved using the initial context. For example the following segment of code retrieves :

```
Topic topic
=(Topic)initContex.lookpu("sbyn.inTopicToSunMicrosystemsTopic");
...
```

Here's an example of how to bind a Java Message Queue object:

```
Queue queue = null;
try {
   queue = newSTCQueue("inQueueToSunMicrosystemsQueue");
   initContext.bind ("sbyn.ToSunMicrosystemQueue", queue);
}
```

```
catch (NameAlreadyBoundException ex) {
  try {
    if(queue != null)
    initContext.rebind ("sbyn.ToSunMicrosystemsQueue", queue);
  }
  catch (Exception ex) {
  throw ex;

  }
}
```

❑ **Check Your Progress – 5 :**

1. JNDI programming uses :

   a. data logic b. web logic

   c. both a and b d. neither a nor b

11.7 Exploring Javax.naming Package :

In Java we see that package is a mixture of classes and interfaces where every package carries its own name which are arranged from top-level classes along with interfaces in different namespace or name collection. Although same-named classes and interfaces cannot appear in similar package but can present in different packages as separate namespace is assigned to every package.

Java.naming package is naming operations of Java Naming and Directory Interface that shows naming and directory functions to applications that are written in Java programming language. Such packages are designed to be independent of any particular naming or directory services working. Hence a variety of services which are new, emerging and previously deployed can access in common way.

Such package shows notion of context given by Context interface having set of name-to-object bindings. Context is the core interface for looking up, binding, unbinding, and renaming objects, and for creating and destroying subcontexts.

In such package, lookup() is commonly used operation that name the object of you choice and in turn returns object with that name. For example, the following code fragment looks up a printer and sends a document to the printer object to be printed :

```
Printer printer = (Printer)ctx.lookup("treekiller");
printer.print(report);
```

Provides the classes and interfaces for accessing naming services.

Interface Summary	
Interface	**Description**
Context	This interface represents a naming context, which consists of a set of name-to-object bindings.
Name	The Name interface represents a generic name -- an ordered sequence of components.
NameParser	This interface is used for parsing names from a hierarchical namespace.
NamingEnumeration<T>	This interface is for enumerating lists returned by methods in the javax.naming and javax.naming.directory packages.
Referenceable	This interface is implemented by an object that can provide a Reference to itself.

Class Summary	
Class	**Description**
BinaryRefAddr	This class represents the binary form of the address of a communications end-point.
Binding	This class represents a name-to-object binding found in a context.
CompositeName	This class represents a composite name -- a sequence of component names spanning multiple name spaces.
CompoundName	This class represents a compound name -- a name from a hierarchical name space.
InitialContext	This class is the starting context for performing naming operations.
LinkRef	This class represents a Reference whose contents is a name, called the link name, that is bound to an atomic name in a context.
NameClassPair	This class represents the object name and class name pair of a binding found in a context.
RefAddr	This class represents the address-of a communications end-point.
Reference	This class represents a reference to an object that is found outside of the naming/directory system.
StringRefAddr	This class represents the string form of the address of a communications end-point.

Exception Summary	
Exception	**Description**
AuthenticationException	This exception is thrown when an authentication error occurs while accessing the naming or directory service.
AuthenticationNonSupportedException	This exception is thrown when the particular flavor of authentication requested is not supported.
CannotProceedException	This exception is thrown to indicate that the operation reached a point in the name where the operation cannot proceed any further.
CommunicationException	This exception is thrown when the client is unable to communicate with the directory or naming service.
ConfigurationException	This exception is thrown when there is a configuration problem.
ContextNotEmptyException	This exception is thrown when attempting to destroy a context that is not empty.
InsufficientResourcesException	This exception is thrown when resources are not available to complete the requested operation.
InterruptedNamingException	This exception is thrown when the naming operation being invoked has been interrupted.
InvalidNameException	This exception indicates that the name being specified does not conform to the naming syntax of a naming system.
LimitExceededException	This exception is thrown when a method terminates abnormally due to a user or system specified limit.
LinkException	This exception is used to describe problems encounter while resolving links.
LinkLoopException	This exception is thrown when a loop was detected will attempting to resolve a link, or an implementation specific limit on link counts has been reached.

MalformedLinkException	This exception is thrown when a malformed link was encountered while resolving or constructing a link.
NameAlreadyBoundException	This exception is thrown by methods to indicate that a binding cannot be added because the name is already bound to another object.
NameNotFoundException	This exception is thrown when a component of the name cannot be resolved because it is not bound.
NamingException	This is the superclass of all exceptions thrown by operations in the Context and Dircontext interfaces.
NamingSecurityException	This is the superclass of security-related exceptions thrown by operations in the Context and Dircontext interfaces.
NoInitialContextException	This exception is thrown when no initial context implementation can be created.
NoPermissionException	This exception is thrown when attempting to perform an operation for which the client has no permission.
NotContextException	This exception is thrown when a naming operation proceeds to a point where a context is required to continue the operation, but the resolved object is not a context.
OperationNotSupportedException	This exception is thrown when a context implementation does not support the operation being invoked.
PartialResultException	This exception is thrown to indicate that the result being returned or returned so far is partial, and that the operation cannot be completed.
ReferralException	This abstract class is used to represent a referral exception, which is generated in response to a *referral* such as that returned by LDAP v3 servers.

ServiceUnavailableException	This exception is thrown when attempting to communicate with a directory or naming service and that service is not available.
SizeLimitExceededException	This exception is thrown when a method produces a result that exceeds a size-related limit.
TimeLimitExceededException	This exception is thrown when a method does not terminate within the specified time limit.

❑ **Check Your Progress – 6 :**

1. Java.naming package is independent of :

   a. naming services b. directory services

   c. both a and b d. neither a nor b

11.8 Let Us Sum Up :

In this unit we have learnt that Java Naming and Directory Interface will show naming and directory functions to applications written in Java programming language.

JNDI is similar to JDBC in that they are both Object-Oriented Java APIs that provide a common abstraction for accessing services from different vendors.

It is noted that fundamental facility in any computing system is naming service--the means by which names are associated with objects and objects are found based on their names.

The Java Naming and Directory InterfaceTM (JNDI) provides naming and directory functionality to applications written in the JavaTM programming language.

JNDI is also defined independent of any specific naming or directory service implementation. It enables applications to access different, possibly multiple, naming and directory services using a common API.

11.9 Answer for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (a)

❑ **Check Your Progress 2 :**

1. (d)

❑ **Check Your Progress 3 :**

1. (a)

❑ **Check Your Progress 4 :**

1. (d)

❑ **Check Your Progress 5 :**

1. (b)

❑ **Check Your Progress 6 :**

1. (c)

11.10 Glossary :

1. **JNDI :** It is Java Naming and Directory Interface which gives name and directory functions to applications in Java programming language.
2. **Directory Object :** It is an object in computing environment used to represent printer, person, computer or network.

11.11 Assignment :

Explain the directory services ?

11.12 Activities :

Study about features of JNDI.

11.13 Case Study :

Steps involved in Programming with JNDI

11.14 Further Readings :

1. Operating System Concept by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne

BLOCK SUMMARY :

In this block, students have learnt and understand about the basic steps involved in deploying RMI Application on same system. The block gives an idea on the study and concept of Beans Development Kit and its development and working criterias. The students have be well explained on the programming concept of JNDI along with Naming and Directory Service.

The block detailed about the basic of RMI Architecture techniques. The concept related to exploring of Javax.naming package along with its features are well explained to the students. The student will be demonstrated practically about writing technique of Simple Bean application.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. What is Beans Development Kit ?
2. Explain the function of JNDI ?
3. Write note on Naming and Directory Service in API ?
4. Write short note on Advanced RMI Concepts ?

❖ **Long Questions :**

1. Write short notes on RMI Architecture ?
2. Write short note on Properties ?
3. Write note on working of RMI ?

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	8	9	10	11
No. of Hrs.				

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....

Dr. Babasaheb Ambedkar
Open University Ahmedabad

BCAR-304

OBJECT ORIENTED CONCEPTS AND PROGRAMMING – II (ADVANCE JAVA)

BLOCK 4 : SERVLETS & JSP PROGRAMMING

SERVLETS & JSP PROGRAMMING

Block Introduction :

Struts is an application development framework that is designed for and used with the popular J2EE (Java 2, Enterprise Edition) platform. JSP is a server side technology which helps to create a webpage dynamically using java as the programming language. In this block, we will detail about the basic concept of Architecture of the Servlet Package. The block will focus on the study and concept of various issues that appears at the time of threading. The students will give an idea on Struts Controller Components and its characteristics features. In this block, the student will made to learn and understand about the basic of about configuring web.xml file for Struts. The concept related to Struts Action Classes and Model Components will also be explained to the students. The student will be demonstrated practically about running Servlets in Apache Tomcat 4.0.

Block Objectives :

After learning this block, you will be able to understand :

- Concept of Architecture of the Servlet Package
- Idea about Threading Issues
- Features about compiling and running Servlets in Apache Tomcat 4.0
- Idea about Struts Controller Components
- Basic of Struts Action Classes and Model Components
- Idea about configuring web.xml file for Struts
- Basic of JSP Syntax and Implicit Objects

Block Structure :

Unit
12

SERVLETS

UNIT STRUCTURE

12.0 Learning Objectives :

After learning this Unit, you will be:

- Able to define servlet and its life cycle
- Able to discuss different types of servlet
- Able to write servlet to serve client request
- Able to execute servlet through tomcat server

12.1 Introduction :

Java Servlet is a platform-independent, container-based Web component used to generate dynamic content in a Web page. It is one of the established technologies to share server-side resources in client-server programming. As Servlet runs in a multi-threaded environment provided by the container, the life cycle events are completely dependent upon its efficient implementation.

The container, also referred to as a servlet engine, is provided by a Web or an application server within which servlets run. Similar to simple Java programming, servlets are Java classes compiled to bytecode. These bytecodes are loaded dynamically into a Java technology-enabled Web server. The servlet classes generally interact with the server via an HTTP request/response mechanism implemented by the servlet engine. The primary function of the container is to contain servlet classes and manage their life cycle.

Java Servlets is web programming technology in Java. A Servlet is used to enhance client-server programming model and develop web applications. Java

Servlet is a part of Java Enterprise Edition (Java EE). The javax.servlet and javax.servlet.http package provides the interfaces and classes for writing Servlet program.

12.2 Servlet Types & Life Cycle :

There are mainly two types of servlets :

1. **Generic Servlet :** Generic servlets extend javax.servlet.GenericServlet. Generic servlet is protocol independent servlet. It implements the Servlet and ServletConfig interface. It may be directly extended by the servlet. Writing a servlet in in GenericServlet is very easy. It has only init() and destroy() method of ServletConfig interface in its life cycle. It also implements the log method of ServletContext interface.
2. **Http Servlet :** HTTP servlets extend javax.servlet.HttpServlet. HttpServlet is HTTP (Hyper Text Transfer Protocol) specific servlet. It provides an abstract class HttpServlet for the developers to extend and create their own HTTP specific servlets. The sub class of HttpServlet must overwrite at least one method out of given below methods :
   - doGet()
   - doPost()
   - doPost()
   - doTrace()
   - doDelete()
   - init()
   - destroy()
   - getServiceInfo()

There is no need to overrride service() method. All the servlet either Generic Servlet or Http Servlet passes there config parameter to the Servlet interface.

Life Cycle of Servlet

A servlet follows a certain life cycle. The servlet life cycle is managed by the servlet container. The life cycle is shown in figure 1.

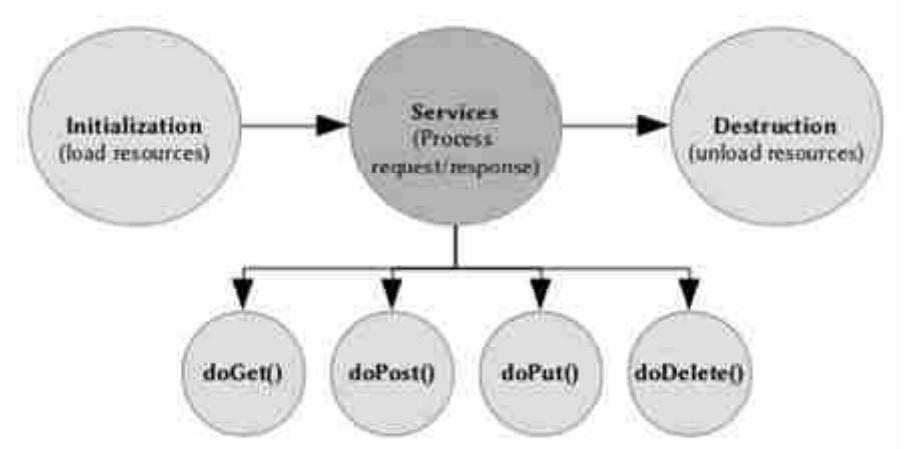


Fig. 1. Servlet Lifecycle

Load Servlet Class

Before a servlet can be invoked the servlet container must first load its class definition. This is done just like any other class is loaded.

Create Instance of Servlet

When the servlet class is loaded, the servlet container creates an instance of the servlet. Typically, only a single instance of the servlet is created, and concurrent requests to the servlet are executed on the same servlet instance. This is really up to the servlet container to decide, though. But typically, there is just one instance.

Call the Servlets init() Method

When a servlet instance is created, its init() method is invoked. The init() method allows a servlet to initialize itself before the first request is processed. We can specify init parameters to the servlet in the web.xml file.

Call the Servlets service() Method

For every request received to the servlet, the servlets service() method is called. For HttpServlet subclasses, one of the doGet(), doPost() etc. methods are typically called. As long as the servlet is active in the servlet container, the service() method can be called. Thus, this step in the life cycle can be executed multiple times.

Call the Servlets destroy() Method

When a servlet is unloaded by the servlet container, its destroy() method is called. This step is only executed once, since a servlet is only unloaded once. A servlet is unloaded by the container if the container shuts down, or if the container reloads the whole web application at runtime.

12.3 Servlet API :

The Servlet API contains two important packages that provide all important classes and interface.

These are as follows :

1. javax.servlet
2. javax.servlet.http

Classes and Interfaces of javax.servlet package :

Interfaces	Classes
Servlet	GenericServlet
ServletConfig	HttpConstraintElement
ServletContext	HttpMethodConstraintElement
ServletContextListner	MultipartConfigElement
ServletRegistration	ServletContextEvent
ServletRequest	ServletInputStream
ServletRequestListner	ServletOutputStream
ServletResponse	ServletRequestAttributeEvent
ServletCookieConfig	ServletRequestEvent

SingleThreadModel	ServletRequestEvent
Filter	ServletRequestWrapper
FilterConfig	ServletResponseWrapper
FilterChain	ServletSecurityElement

Classes and Interfaces of javax.servlet.http package :

Interfaces	Classes
HttpServletRequest	Cookie
HttpServletResponse	HttpServlet
HttpSession	HttpServletRequestWrapper
HttpSessionActivationListner	HttpServletResponseWrapper
HttpSessionContext	HttpSessionBindingEvent
HttpSessionAttributeListner	HttpSessionEvent
HttpSessionIdListner	HttpUtils

Servlet Interface

Servlet interface defines some methods that all the servlet classes must implement. This method provides the following five methods. Out of these five methods, three methods are Servlet life cycle methods.

Servlet Interface Methods

Following are the methods of Servlet Interface :

Methods	Description
public void init(ServletConfig, config)	It is one of the Servlet life cycle methods. It is invoked by Servlet container after being initialized by Servlet.
public void service (ServletRequest req, ServletResponse res)	The **service**( ) method is called after successful completion of **init**( ). It is invoked by Servlet container to respond to the requests coming from the client.
public ServletConfig getServletConfig ( )	Returns a ServletConfig object, which contains initialization and startup parameters for this Servlet.
public String getServletInfo( )	Returns the information about the Servlet, such as author, version, and copyright. This method returns a string value.
public void destroy( )	Called by servlet container and it marks the end of the life cycle of a servlet. It indicates that servlet has been destroyed.

HttpServlet Class

The HttpServlet class extends the GenericServlet and implements Serializable interface. It is an abstract class. The HttpServlet class reads the HTTP request from http, get, post, put, delete etc. It calls one of the corresponding methods.

HttpServlet Class Methods

- protected void doGet(HttpServletRequest req, HttpServletResponse resp)
- protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
- protected void doHead(HttpServletRequest req, HttpServletResponse resp)
- protected void doPost(HttpServletRequest req, HttpServletResponse resp)
- protected void doPut(HttpServletRequest req, HttpServletResponse resp)
- protected void doTrace(HttpServletRequest req, HttpServletResponse resp)
- protected void service(HttpServletRequest req, HttpServletResponse resp)

GenericServlet Class

It is an abstract class that implements Servlet, ServletConfig and serializable interface. It provides the implementation of all methods of these interfaces except the service method.

GenericServlet may be directly extended by Servlet. It provides simple versions of the life cycle methods init( ) and destroy( ) methods.

GenericServlet Class Methods

Following are the important methods of GenericServlet Class :

Methods	Description
public void destroy( )	Invoked by servlet container. It shows that the servlet is being taken out of service.
public String getInitParameter(String name)	Returns a String containing the value of named parameter.
public String getServletInfo( )	Returns information related to Servlet like author, version etc.
public String getServletName( )	Returns the name of Servlet object.
public void init( )	It is a convenience method that can easily be overridden so that we do not need to call **super.init(config).**
public void log(String msg)	Writes the given message to a Servlet log file.
public abstract void service(ServletRequest req, ServletResponse res)	It is an abstract method, called by the servlet container to allow the servlet to respond to a request.

❑ **Check Your Progress – 1 :**

1. What is the lifecycle of a servlet ?
   a. Servlet class is loaded
   b. Servlet instance is created
   c. init,Service,destroy method is invoked
   d. All mentioned above
2. Which packages represent interfaces and classes for servlet API ?
   a. javax.servlet
   b. javax.servlet.http
   c. Both a & b
   d. None of the above

12.4 Threading Issues :

A thread is many times called an execution context or a lightweight process which is single sequential flow of control within a program. When you run one of these sorting applets, it creates a thread that performs the sort operation. Each thread is a sequential flow of control within the same program. Each sort operation runs independently from the others, but at the same time.

When we say that a program is multithreaded, we are not implying that the program runs two separate instances simultaneously. Rather, we are saying that the same instance spawns multiple threads that process this single instance of code. This means that more than one sequential flow of control runs through the same memory block.

When multiple threads execute a single instance of a program and therefore share memory, multiple threads could possibly be attempting to read and write to the same place in memory. In fig 1.2 we see a multithreaded program where multiple threads processing are done at same time.

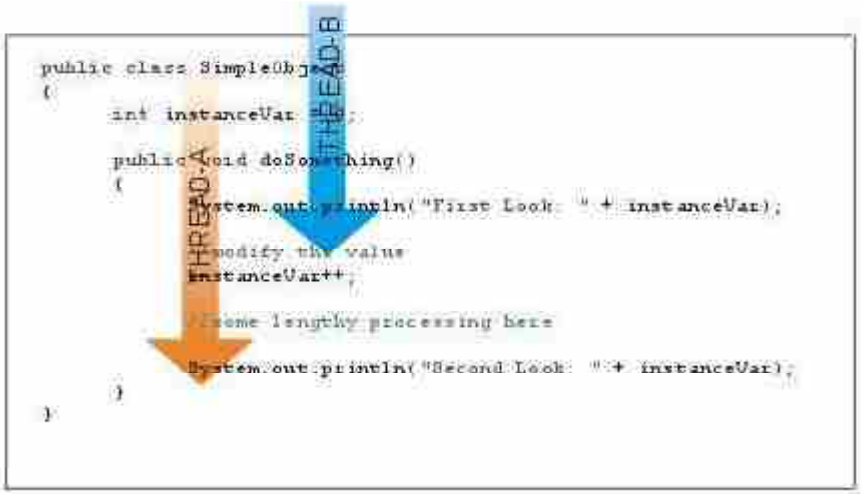


Fig 1.2 multiple threads processing

In this we see that when thread A analyse variable instanceVar, then it is noted that thread-B just increases instanceVar. We see that the problem here appears as thread A written to instanceVar which does not expect value to change till thread A clearly performs. Fortunately thread-B also analyse same thing itself; the only problem is they share the same variable. So such issue is not exclusive to servlets, but serves as common programme problem in case of multithreading application.

So we see that Servlet instances are essentially not thread safe for the reason of multi threading nature of Java programming language. The Java Virtual Machine supports working similar code by multiple threads. This is a great performance benefit on machines which have multiple processors. This also allows the same code to be executed by multiple concurrent users without blocking each other.

❑ **Check Your Progress – 2 :**

1. Servlets handle multiple simultaneous requests by using threads.

   a. True b. False

12.5 Session Tracking :

HTTP protocol and Web Servers are stateless, what it means is that for web server every request is a new request to process and they can't identify if it's coming from client that has been sending request previously.

But sometimes in web applications, we should know who the client is and process the request accordingly. For example, a shopping cart application should know who is sending the request to add an item and in which cart the item has to be added or who is sending checkout request so that it can charge the amount to correct client.

Session is a conversional state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response.

There are several ways through which we can provide unique identifier in request and response.

1. **User Authentication :** This is the very common way where we user can provide authentication credentials from the login page and then we can pass the authentication information between server and client to maintain the session. This is not very effective method because it won't work if the same user is logged in from different browsers.
2. **HTML Hidden Field :** We can create a unique hidden field in the HTML and when user starts navigating, we can set its value unique to the user and keep track of the session. This method can't be used with links because it needs the form to be submitted every time request is made from client to server with the hidden field. Also it's not secure because we can get the hidden field value from the HTML source and use it to hack the session.
3. **URL Rewriting :** We can append a session identifier parameter with every request and response to keep track of the session. This is very tedious because we need to keep track of this parameter in every response and make sure it's not clashing with other parameters.
4. **Cookies :** Cookies are small piece of information that is sent by web server in response header and gets stored in the browser cookies. When client make further request, it adds the cookie to the request header and we can utilize it to keep track of the session. A cookie has a name, a single value, expiration date and optional attributes. A cookie's value can uniquely identify a client. Since a client can disable cookies, this

is not the most secure and fool-proof way to manage the session. We can maintain a session with cookies but if the client disables the cookies, then it won't work. If Cookies are disabled then we can fallback to URL rewriting to encode Session id e.g. JSESSIOINID into the URL itself.

5. **Session Management API :** Session Management API is built on top of above methods for session tracking. HttpSession object is used to store entire session with a specific client. We can store, retrieve and remove attribute from HttpSession object. Any servlet can have access to HttpSession object throughout the getSession() method of the HttpServletRequestobject.

Some of the major disadvantages of all the above methods are :

- Most of the time we don't want to only track the session, we have to store some data into the session that we can use in future requests. This will require a lot of effort if we try to implement this.
- All the above methods are not complete in themselves, all of them won't work in a particular scenario. So we need a solution that can utilize these methods of session tracking to provide session management in all cases.

❑ **Check Your Progress – 3 :**

1. Which method in session tracking is used in a bit of information that is sent by a web server to a browser and which can later be read back from that browser ?

a. HttpSession b. URL rewriting

c. Cookies d. Hidden form fields

12.6 Writing and Running Servlets in Apache Tomcat 7 :

To create a Servlet application we need to follow the below mentioned steps. These steps are common for all the Web server. Apache Tomcat is an open source web server for testing servlets and JSP technology.

- Create directory structure for the application
- Create a Servlet
- Compile the Servlet
- Create the deployment descriptor of the application
- Start the server and deploy the application

Creating the Directory Structure

There is a unique directory structure that must be followed to create Servlet application. This structure tells where to put the different types of files.

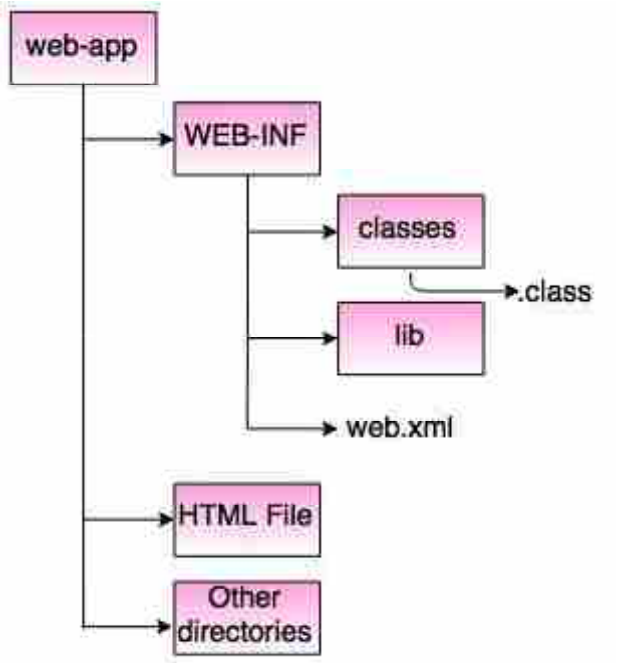


Fig 1.3 Directory Structure of Servlet Application

Create a Servlet

```
//ServletTest.java
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class ServletTest extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
    {
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();
        pw.println("<html><body>");
        pw.println("Welcome to BAOU, Ahmedabad");
        pw.println("</body></html>");
        pw.close();
    }
}
```

Compile the Servlet program

Assuming the classpath and environment is setup properly.

Create a deployment descriptor

The deployment descriptor is an xml file. It is used to map URL to servlet class, defining error page.

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_9" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
<display-name>Servlet Test</display-name>
  <servlet>
    <servlet-name>World</servlet-name>
    <servlet-class>ServletTest</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>World</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>
</web-app>
```

Save the web.xml file in <Tomcat-installation directoryt>/webapps/ROOT/WEB-INF/

- Now start the Tomcat server
- Open browser and type

  http://localhost:8080/BAOU/hello
- It will execute our servlet and display following output.

❑ **Check Your Progress – 4 :**

1. Which method is used to specify before any lines that uses the PintWriter ?

   a. setPageType() b. setContextType()

   c. setContentType() d. setResponseType()

2. What type of servlets use these methods doGet(), doPost(),doHead, doDelete() and doTrace() ?

   a. Genereic Servlets b. HttpServlets

   c. All of the above d. None of the above

12.7 Request Dispatcher :

The RequestDispatcher interface defines an object that receives the request from client and dispatches it to the resource(such as servlet, JSP, HTML file).

This interface has following two methods :

- public void forward(ServletRequest request, ServletResponse response): It forwards the request from one servlet to another resource (such as servlet, JSP, HTML file).
- public void include(ServletRequest request, ServletResponse response): It includes the content of the resource(such as servlet, JSP, HTML file) in the response.

Difference between forward() vs include() :

To understand the difference between these two methods, lets take an example :

Suppose we have two pages X and Y. In page X we have an include tag, this means that the control will be in the page X till it encounters include tag, after that the control will be transferred to page Y. At the end of the processing of page Y, the control will return back to the page X starting just after the include tag and remain in X till the end.

In this case the final response to the client will be send by page X.

Now, we are taking the same example with forward. We have same pages X and Y. In page X, we have forward tag. In this case the control will be in page X till it encounters forward, after this the control will be transferred to page Y. The main difference here is that the control will not return back to X, it will be in page Y till the end of it. In this case the final response to the client will be send by page Y.

Getting RequestDispatcher

RequestDispatcher can be obtained from a request object or from a servlet context.

- RequestDispatcher dispatcher = request.getRequestDispatcher("ved.jsp");

  dispatcher.forward(request, response);

We can get the RequestDispatcher from the request object with the getRequestDispatcher() method.

- RequestDispatcher dispatcher = getServletContext().getRequestDispatcher ("/ved.jsp");

  dispatcher.forward(request, response);

Here we get the RequestDispatcher from the servlet context. In this case, the path must begin with a slash character.

Example :

In this example, we are using both the methods include and forward. Using include method, we will be changing the content of current page and when we are ready to transfer the control to the next page, we will use forward method.

index.html:

```
<html>
<form action="loginPage" method="post">
   User Name:<input type="text" name="uname"/><br/>
   Password:<input type="password" name="upass"/><br/>
```

```
    <input type="submit" value="SUBMIT"/>
</form>
</html>
```

Validation.java:

```
import java.io.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Validation extends HttpServlet
{
    public void doPost(HttpServletRequest request,
        HttpServletResponse response)
            throws ServletException, IOException
        {
            response.setContentType("text/html");
            PrintWriter pw = response.getWriter();
            String name=request.getParameter("uname");
            String pass=request.getParameter("upass");
            if(name.equals("ved") &&  pass.equals("desai"))
            {
                RequestDispatcher dis=request.getRequestDispatcher
("welcome");
                dis.forward(request, response);
            }
            else
            {
                pw.print("User name or password is incorrect!");
                RequestDispatcher dis=request.getRequestDispatcher
("index.html");
                dis.include(request, response);
            }
        }
    }
```

WelcomeTest.java:

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

```
public class WelcomeTest extends HttpServlet {
    public void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException
    {

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

        String name=request.getParameter("uname");
        pw.print("Hello"+name+"!");
        pw.print(" Welcome to BAOU");
}
}
```

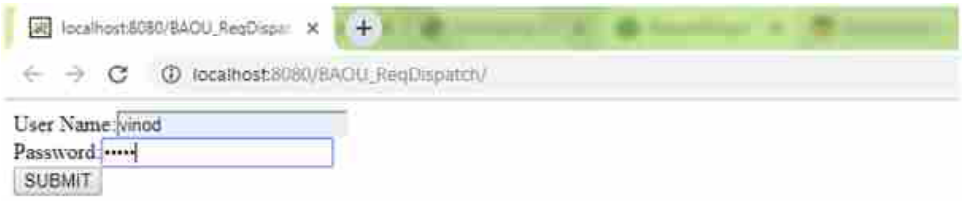
web.xml :

```
<web-app>
    <display-name>BAOU-Ahmedabad</display-name>
    <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    </welcome-file-list>
        <servlet>
    <servlet-name>Login</servlet-name>
    <servlet-class>Validation</servlet-class>
    </servlet>
    <servlet>
    <servlet-name>Welcome</servlet-name>
    <servlet-class>WelcomeTest</servlet-class>
    </servlet>
    <servlet-mapping>
    <servlet-name>Login</servlet-name>
    <url-pattern>/loginPage</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
    <servlet-name>Welcome</servlet-name>
    <url-pattern>/welcome</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
    <welcome-file>index.html</welcome-file>
```

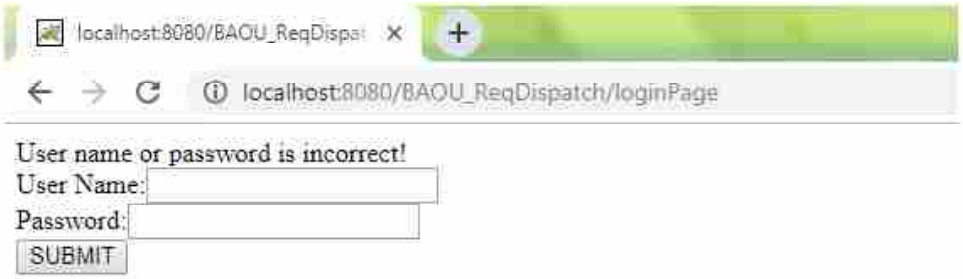
```
</welcome-file-list>
</web-app>
```

Output :

Entering wrong credentials :



Error screen :



Welcome screen on entering correct user name and password:

❑ Check Your Progress – 5 :

1. Which method is used to send the same request and response objects to another servlet in RequestDispacher ?

   a. forward() b. sendRedirect()

   c. Both A & B d. None of the above

12.8 Let Us Sum Up :

In this unit we have learnt that Servlets are modules of Java code that run in a server application to answer client requests.

In order to initialize a Servlet, a server application loads the Servlet class and creates an instance by calling the no-args constructor.

An HTTP Servlet handles client requests through its service method. The service method supports standard HTTP client requests by dispatching each request to a method designed to handle that request.

A thread is many times called an execution context or a lightweight process which is single sequential flow of control within a program. When you run one of these sorting applets, it creates a thread that performs the sort operation.

Cookies serve as a facility for servers to send information to a client. This information is then housed on the client, from which the server can later retrieve the information.

RequestDispatcher is an interface, implementation of which defines an object which can dispatch request to any resources on the server.

12.9 Answer for Check Your Progress :

- **Check Your Progress 1 :**

  1. (d), **2.** (c)

- **Check Your Progress 2 :**

  1. (a)

- **Check Your Progress 3 :**

  1. (c)

- **Check Your Progress 4 :**

  1. (c), **2.** (b)

- **Check Your Progress 5 :**

  1. (a)

12.10 Glossary :

1. **HTTP :** It is the data communication protocol used to establish communication between client and server.
2. **Container :** It is used in java for dynamically generating the web pages on the server side.
3. **Content-Type :** It is HTTP header that provides the description about what are you sending to the browser.

12.11 Assignment :

1. Write short note on different Http Methods.
2. Discuss serveltConfig and servletContext.
3. Differentiate between Get and Post method.

12.12 Activities :

Collect some information on Http Protocol.

12.13 Case Study :

Anatomy of an HTTP GET and POST request.

12.14 Further Readings :

1. https://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html
2. https://www.studytonight.com/servlet/
3. https://www3.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html

Unit 13

INTRODUCTION TO STRUTS

UNIT STRUCTURE

13.0 Learning Objectives :

After learning this Unit, you will be:

- Able to define Struts Action Classes
- Able to explain Struts Model Components
- Able to explain The Struts View Components

13.1 Introduction :

Today's world is "digital world" where people are highly dependent on the internet and web applications. Millions of web applications and thousands of technologies are available in the market for enterprise application development. All of these web technologies are categorized as Model1, Model2, MVC and much more. Strut is MVC Model2 architecture based framework. Throughout this chapter we have discussed all concepts with respect to struts 2.

MVC means Model-View-Controller where model is responsible for business logic, view handles the user interaction with application and controller decides the flow of application.

Struts is an application development framework that is designed for and used with the popular J2EE (Java 2, Enterprise Edition) platform. It cuts time out of the development process and makes developers more productive by providing them a series of tools and components to build applications with. It is non-proprietary and works with virtually any J2EE-compliant application server. Struts falls under the Jakarta subproject of the Apache Software Foundation

and comes with an Open Source license (meaning it has no cost and its users have free access to all its internal source code).

ActionServlet is the class that plays role of controller in Struts. Whenever the user sends a request to server it passes via ActionServlet class. It then decides the necessary model and sends this request to the respective model. Two more files support the execution of application: web.xml and struts-config.xml files. Web.xml file is deployment descriptor which keeps all the application related settings while struts-config.xml file maps a request to Action classes and ActionForms (a simple POJO which contains properties related to UI). Model is handled by various Java technologies like EJB, JDBC, Hibernate, Spring etc. It mainly concentrates on the business logic and semantics of the application. While view can be developed using JSP, Velocity Templates, JSTL, OGNL, XSLT and other HTML technologies. View is responsible for getting the input from the user and rendering the result of that input sent back to user.

13.2 Life Cycle of Struts Request and it's Controller Components :

Life Cycle of Struts 2 consists of :

1. When a request comes web container maps the request in the web.xml and calls the controller (FilterDispatcher).
2. FilterDispatcher calls the ActionMapper to find an Action to be invoked.
3. FilterDispatcher calls the ActionProxy.
4. ActionProxy reads the information of action and interceptor stack from configuration file using configuration manager (struts.xml) and invoke the ActionInvocation.
5. ActionInvocation calls the all interceptors one by one and then invoke the action to generate the result.
6. action is executed ActionInvocation again calls the all interceptors in reverse order.
7. Control is returned to the FilterDispatcher.
8. Result is sent to the client.

The controller is responsible for intercepting and translating user input into actions to be performed by the model. The controller is responsible for selecting the next view based on user input and the outcome of model operations. The Controller receives the request from the browser, invoke a business operation and coordinating the view to return to the client. The controller is implemented by a java servlet, this servlet is centralized point of control for the web application. In struts framework the controller responsibilities are implemented by several different components like :

1. The ActionServlet Class
2. The RequestProcessor Class
3. The Action Class

Controller components coordinate activities in the application. This may mean taking data from the user and updating a database through a Model component, or it may mean detecting an error condition with a back-end system and directing the user through special error processing. Controller components

accept data from the users, decide which Model components need to be updated, and then decide which View component needs to be called to display the results.

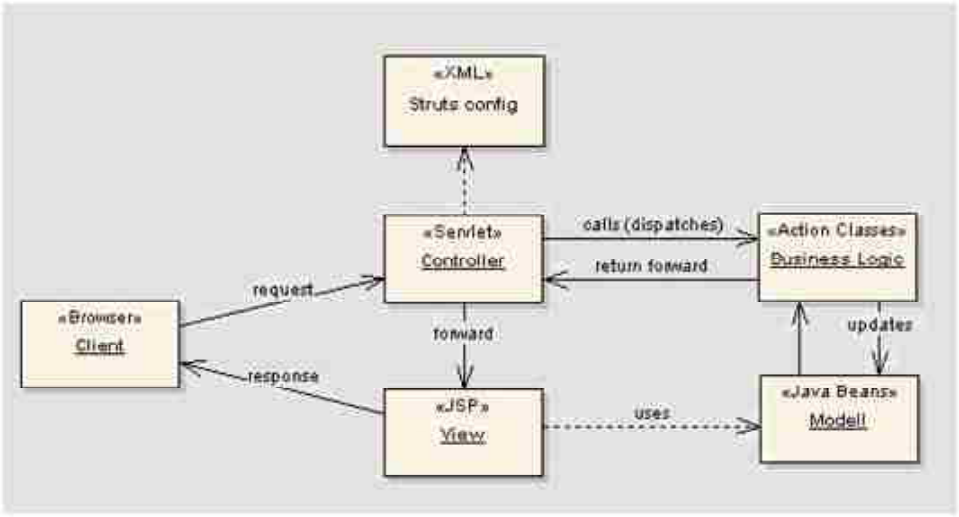


Fig 2.1 Controller component

One of the major contributions of Controller components is that they allow the developer to remove much of the error handling logic from the JSP pages in their application. This can significantly simplify the logic in the pages and make them easier to maintain.

Controller maps the user request to specific action. In Struts 2; StrutsPrepareAndExecuteFilter acts as Controller. Controller receives the user request and decides which action to be invoked. It creates an instance of this action; invoke interceptors (if any). Then it calls invoke() method of ActionInvocations that executes the action. ActionInvocation then calls the intercept() method. The intercept() method of ActionInvocation class in turn calls the invoke() method of the ActionInvocation till all the interceptors are invoked. Interceptor are invoked before and after the action is executed. Interceptors are executed in the order they are defined in the stack. Finally action itself is invoked and the result is generated. All the interceptors are then invoked again but this time in the reverse order.

In the above figure we see that when user sends a query with a browser, then the controller (servlet) gets and processes such query and decides which action will be called or which view component this query should be forward. Once the controller calls an action, action can read data from database and shows data to the model component, java beans. The action returns next step to controller which further will check what kind is the next step. (JSP View, next action, …) and forwards to it.

View displays the result. View can be either JSP page, Velocity templates, XSLT pages, Freemaker or some other presentation-layer technology. Object-Graph Navigation Language (OGNL) is used to reference and manipulate data on the ValueStack.

❑ **Check Your Progress – 1 :**

1. Struts supports which of these model components ?

   a. JavaBeans b. EJB c. CORBA

   d. JDO e. All of these

2. ActionServlet, RequestProcessor and Action classes are the components of

a. View b. Model c. Controller d. Deployment

13.3 Struts Action Classes :

It is seen that actions are the main part of Struts2 framework since they are for any Model View Controller structure. Every URL is mapped to particular action that shows processing logic required to service request from user. In this, action also serves in two capacities:

In any struts 2 application, there is an action class associated with different type of client action. There are four ways through which we can create Struts 2 Action classes.

1. Simple Action Class:

We can use any normal java class as Struts 2 action class, the only requirement is that it should have execute() method returning String.

2. Using Struts 2 Annotations:

Struts 2 supports annotation based configuration and we can use it to create action classes.

3. Extending ActionSupport Class:

ActionSupport class is the default implementation of Action interface and it also implements interfaces related to Validation and i18n support. ActionSupport class implements Action, Validateable, ValidationAware, TextProvider and LocaleProvider interfaces.

4. Implementing Action interface:

We can implement com.opensymphony.xwork2.Action interface also to create action classes. Action interface has a single method execute() that we need to implement. The only benefit of using Action interface is that it contains some constants that we can use for result pages, these constants are SUCCESS, ERROR, NONE, INPUT and LOGIN. We can rewrite above HomeAction class by implementing Action interface as shown below.

```
public interface Action
{
   public static final String SUCCESS = "success";
   public static final String NONE = "none";
   public static final String ERROR = "error";
   public static final String INPUT = "input";
   public static final String LOGIN = "login";
   public String execute() throws Exception;
}
```

Action interface contains only one method execute that should be implemented overridden by the action class even if we are not forced. In the action method shown below with HelloTestAction, we can see that :

```
public class HelloTestAction
{
   private String userName;
   public String execute() throws Exception {
      return "success";
   }
   public String getuserName () {
      return userName;
   }
   public void setuserName (String name) {
      userName = name;
   }
}
```

To explain point used by action method to controls the view, the following changes are made to carry out method and extend the class ActionSupport as follows :

```
import com.opensymphony.xwork2.ActionSupport;
public class HelloTestAction extends ActionSupport
{
   private String userName;
   public String execute() throws Exception {
      if ("BAOU".equals(userName)) {
         return SUCCESS;
      } else {
         return ERROR;
      }
   }
      public String getuserName() {
      return userName;
   }
   public void setuserName(String name) {
      userName = name;
   }
}
```

Here we see that there appears some logic in execute method which is the name attribute that equals to string BAOU what we return SUCCESS as output, and if not, then it will return ERROR.

❑ **Check Your Progress – 2 :**

1. What does validate() method of ActionForm returns ?

   a. ActionErrors b. ActionForward

   c. ActionMapping d. ActionError

2. Which of the following methods is overridden by Action class ?

   a. run() b. destroy( ) c. execute() d. service()

13.4 Struts Model Components :

The model shows business data for an application that closely resembles real-world entities and business processes for organization. The model components of an application possibly are the most expensive software artefacts' to an organization. The model take account of business entities in addition to the rules that preside over access to furthermore modification of the data. It's vital that this be kept in a single location in order to maintain valid data integrity, reduce redundancy, and increase reusability.

The Struts Framework has no built-in support for the Model layer. The model should remain independent of the type of client that's being used to access the business objects and their associated rules.

In struts, model describes an application's data having logic for accessing and manipulating data. In this, any data which is part of persistent state of the application be kept inside the model objects. The business objects update the application state. Action Form bean represents the Model state at a session or request level, and not at a persistent level. It is noted that model services are accessed by controller for querying or effecting changes in model state. The model notifies view when a state change occurs in the model. The JSP file reads information from the ActionForm bean using JSP tags.

The Struts structure doen't offer much in way of creating model components. It is the Enterprise JavaBeans (EJB), Java Data Objects(JDO) and JavaBeans which uses as model. Struts structure doesn't restrict to single particular model implementation.

Struts framework helps for developing the web based applications. Struts java framework is one of the most popular framework for web based applications. Java servlet, JavaBeans, ResourceBundles and XML etc are the Jakarta commons packages used for accomplishing this purpose. This is an open source implementation of MVC pattern for the development of web based application. The features of this type of framework are, more robust or reliable architecture Helps for development of application of any size Easy to design Scalable Reliable web application with java.

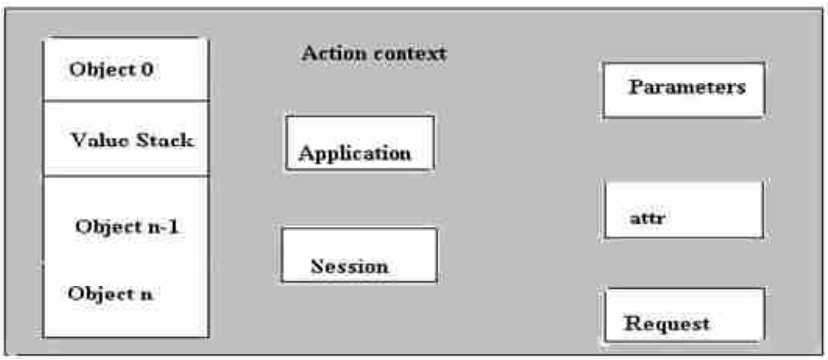


Fig 2.2 Framework

It is noted that MVC implementation by action, result and Filter Dispatcher. Here the controller's work is to map user request to required action which is done by Filter Dispatcher. It includes data and business logic and model is worked out using action component. The presentation component of the MVC pattern is view and view is implemented using JSP, Velocity Template, Freemaker or some other presentation-layer technology.

❑ **Check Your Progress – 3 :**

1. What is return type of execute() method ?

   a. String b. int c. ActionForward

   d. ActionMapping e. ActionForm

13.5 The Struts View Components :

We see that view is responsible for rendering state of model. In this, the presentation semantics are summarize inside the view, as a result of which the model data can be adapted for various different kinds of clients. The view modifies itself when change in model is communicated to view. A view forwards user input to the controller. The view is simply as JSP or HTML file. There is no flow logic, no business logic, and no model information -- just tags. Tags are one of the things that make Struts unique compared to other frameworks like Velocity. The view components typically employed in a Struts application are:

- HTML
- Data transfer objects
- Struts ActionForms
- JavaServer Pages
- Custom tags
- Java resource bundles

Struts ActionForm

Struts ActionForm objects are applied in structure so as to pass client input data back and forth among user and business layer. The structure directly will collect the input from request and pass it to Action with the help of form bean, further which pass on to business layer. To keep the presentation layer decoupled from the business layer, we should not pass the ActionForm itself to the business layer; rather, create the appropriate DTO using the data from the ActionForm.

- Java class org.apache.struts.action.ActionForm, which we subclass to create a form bean that is used in two ways at run time:
  - When a JSP page prepares the related HTML form for display, the JSP page accesses the bean, which holds values to be placed into the form. Those values are provided from business logic or from previous user input.
  - When user input is returned from a web browser, the bean validates and holds that input either for use by business logic or (if validation failed) for subsequent redisplay.
- Numerous custom JSP tags that are simple to use but are powerful in the sense that they hide information. Page Designer does not need to

know much about form beans, for example, beyond the bean names and the names of each field in a given bean.

- It maintains the session state for web application.
- The ActionForm object is populates automatically on the server side when data is entered on a client side.

There are three main components that involve in handling form in Struts application: Struts form, JavaBean and Action class. The relationship between these components is that fields in the form will be mapped to properties of a JavaBean which is an attribute of the action class that handles the form submission. In Struts, we don't need to read values of the form's fields through a HTTP request object like in traditional JSP/Servlet. Instead, Struts will automatically fetch values of form's fields into the mapped JavaBean object. Then in the action class, we can access the form's fields just like accessing JavaBean properties.

❑ **Check Your Progress – 4 :**

1. In interceptor which is used to display the intermediate result ?

   a. Params Interceptor b. Custom Interceptor

   c. ExecAndWait Interceptor d. Prepare Interceptor

13.6 Configuring the web.xml file for Struts :

The web.xml configuration file is a J2EE configuration file that determines how elements of the HTTP request are processed by the servlet container. It is not strictly a Struts2 configuration file, but it is a file that needs to be configured for Struts2 to work.

The web.xml web application descriptor file represents the core of the Java web application, so it is appropriate that it is also part of the core of the Struts framework. In the web.xml file, Struts defines its FilterDispatcher, the Servlet Filter class that initializes the Struts framework and handles all requests. This filter can contain initialization parameters that affect what, if any, additional configuration files are loaded and how the framework should behave.

```
<?xml version = "1.0" Encoding = "UTF-8"?>
<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xmlns = "http://java.sun.com/xml/ns/javaee"
   xmlns:web = "http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
   xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
   http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
   id = "WebApp_ID" version = "3.0">

   <display-name>Learning Struts 2</display-name>
   <welcome-file-list>
      <welcome-file>index.jsp</welcome-file>
   </welcome-file-list>
```

```
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>
            org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
</filter>
<filter-mapping>
     <filter-name>struts2</filter-name>
     <url-pattern>/*</url-pattern>
</filter-mapping>
</web-app>
```

This sets up a filter that sends every URL to the Struts framework. This is how Struts "sits between" the server and our code. Now when the server receives a request, it will send that request to Struts, and Struts will then send it to our action classes.

The struts.xml file contains the configuration information that we will be modifying as actions are developed. This file can be used to override default settings for an application, for example struts.devMode = false and other settings which are defined in property file. This file can be created under the folder WEB- INF/classes.

❑ **Check Your Progress – 5 :**

1. Struts combines which of these in to a unified Framework ?

   a. Java Servlets

   b. Java Server pages

   c. Custom tags and Message Resources

   d. All of the above

13.7 Writing and Executing Struts Application :

Struts application is a normal web application that carries Struts libraries and configuration files. Such application can be created in similar manner as we creates any other web application.

Example :

First, Create a Java package called "example" under C:\tomcat8\webapps\ Login\WEB-INF\ classes directory.

Then, Copy the following important jar files from struts-2.x.x\lib directory to project's Login\WEB-INF\lib directory:

- commons-fileupload-1.2.2.jar
- commons-io-2.0.1.jar
- commons-lang3-3.1.jar
- commons-logging-1.1.1.jar
- commons-logging-api-1.1.jar
- freemarker-2.3.19.jar

- javassist-3.11.0.GA.jar
- ognl-3.0.6.jar
- struts2-core-2.3.8.jar
- xwork-core-2.3.8.jar

These are core Struts libraries which are necessary for running our application.

Now, Create **index.jsp** file under project's Login directory with the following code:

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%-- Using Struts2 Tags in JSP --%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Login Page</title>
</head>
<body>
<h3>Welcome User, please login below</h3>
<s:form action="login">
        <s:textfield name="name" label="User Name"></s:textfield>
        <s:password name="pwd" label="Password"></s:password>
        <s:submit value="Login"></s:submit>
</s:form>
</body>
</html>
```

The Struts tags are imported by the following taglib directive:

```
<%@ taglib uri="/struts-tags" prefix="s"%>
```

On submitting this form, the action login will be invoked. We will create the action class right now, and configure it in struts.xml file later.

Under package example, create a class called **Login.java** with the following code:

```
package example;
import com.opensymphony.xwork2.Action;
public class Login implements Action
{
   @Override
   public String execute() throws Exception
```

```
    {
        if("vinod".equals(getName()) && "admin".equals(getPwd()))
            return "SUCCESS";
        else
            return "ERROR";
    }

    //Java Bean to hold the form parameters
    private String name;
    private String pwd;
    public String getName() {
            return name;
    }
    public void setName(String name) {
            this.name = name;
    }
    public String getPwd() {
            return pwd;
    }
    public void setPwd(String pwd) {
            this.pwd = pwd;
    }
}
```

In this action class, we declare three private variables name, pwd and its corresponding getters and setters. The framework we will use the setters to fetch values from the input page, and use the getters to print values in result page.

The method execute() is the action method which will be invoked by the framework when the action is called. The method to be invoked will be configured in struts.xml file. This method simply checks login details and returns a String which is logical name of a view. The framework will look for a matching view file and send it to the client. The method returns a String constant named SUCCESS which equals to the logical name "success" and ERROR means "fail".

The mapping of logical view names to physical view files for action classes are configured in struts.xml file.

Now, create **welcome.jsp** file under project's Login directory with the following code :

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Welcome Page</title>
</head>
<body>
<h3>Welcome <s:property value="name"></s:property></h3>
</body>
</html>
```

welcome.jsp page will display the welcome message describing User Name using Struts' <s:property> tag.

Now, create **error.jsp** file under project's Login directory with the following code :

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Error Page</title>
</head>
<body>
<h4>User Name or Password is wrong</h4>
<s:include value="index.jsp"></s:include>
</body>
</html>
```

error.jsp page will display the "User Name or Password is wrong" message and redirects user to index.jsp for re-login using struts <s:include> tag.

Up to this, we have created the input page, response page, the action class and the result page, but they haven't been connected. So we are going to connect these components together in struts.xml configuration file.

Create **struts.xml** file under Login\WEB-INF\classes directory with the following content:

The <package> element defines a logical group of a Struts application. It extends the struts-default package which is built into Struts core to provide some defaults from which other packages can inherit.

The <action>element connects an action class with its views. We specify some attributes:

- name: name of the action. This name must match value of the action's attribute of <form> tag.
- class: fully-qualified name of the action class.
- method: name of a method in the action class (action method). The specified method will be executed by the framework when the action is called.

The <result> elements specify mapping between logical view names (returned from the action method) to physical view files. The framework picks up a view based on the String returned from the action method, which should match with the name attribute of a <result>element. Here we defined two mappings:

- "success" maps to "/welcome.jsp" file: The action method execute() returns "SUCCESS", so the welcome.jsp file will be picked up and sent to the client.
- "error" maps to "/error.jsp" file: The action method execute() returns "ERROR", so the error.jsp file will be picked up and user will be redirected to re-login with proper message.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">
<struts>
<package name="default" extends="struts-default">
    <action name="login" class="example.Login">
        <result name="SUCCESS">welcome.jsp</result>
        <result name="ERROR">error.jsp</result>
        </action>
</package>
</struts>
```

Now, configure dispatcher filter for Struts in web.xml as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_9" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
```

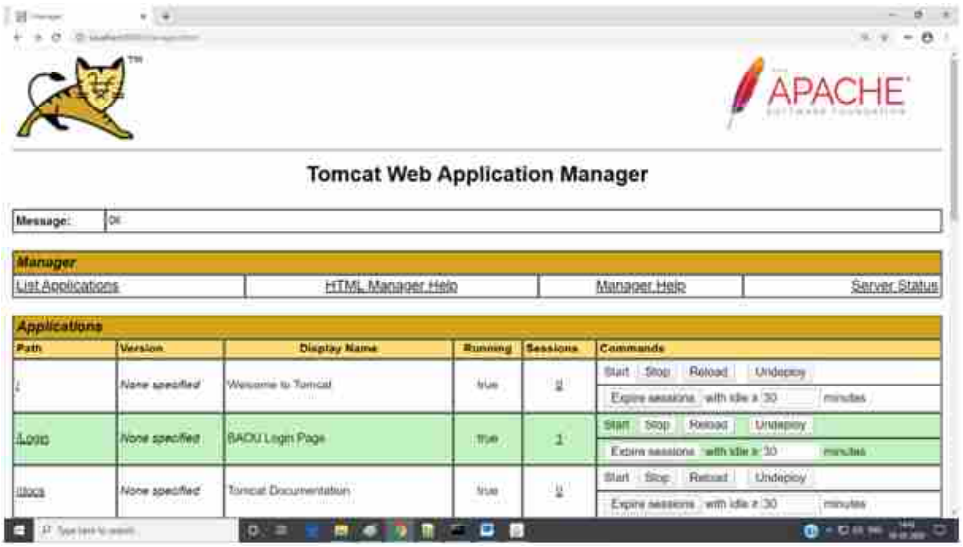
```
    <display-name> BAOU Login Page</display-name>
    <filter>
        <filter-name>struts2</filter-name>
        <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-
class>
    </filter>
        <filter-mapping>
            <filter-name>struts2</filter-name>
            <url-pattern>/*</url-pattern>
        </filter-mapping>
</web-app>
```

In the <filter-mapping> element, we specify that all URLs (pattern: / *) will be handled by the Struts dispatcher filter.

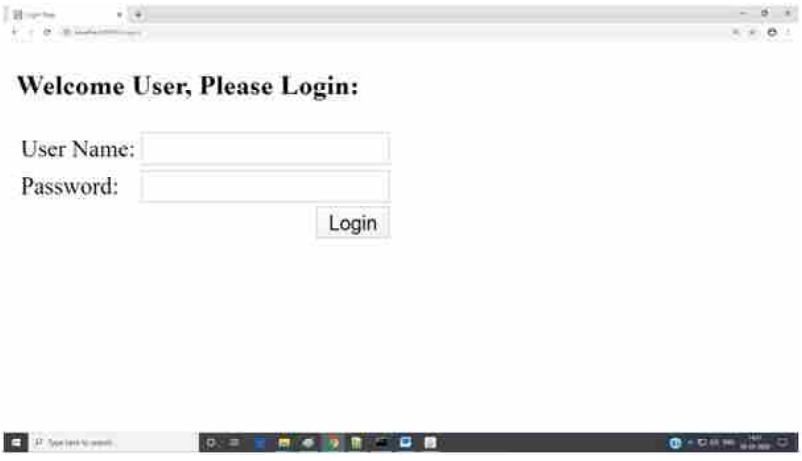
Now, Start the tomcat server and open a web browser. Type the following URL into its address bar:

http://localhost:8080/html

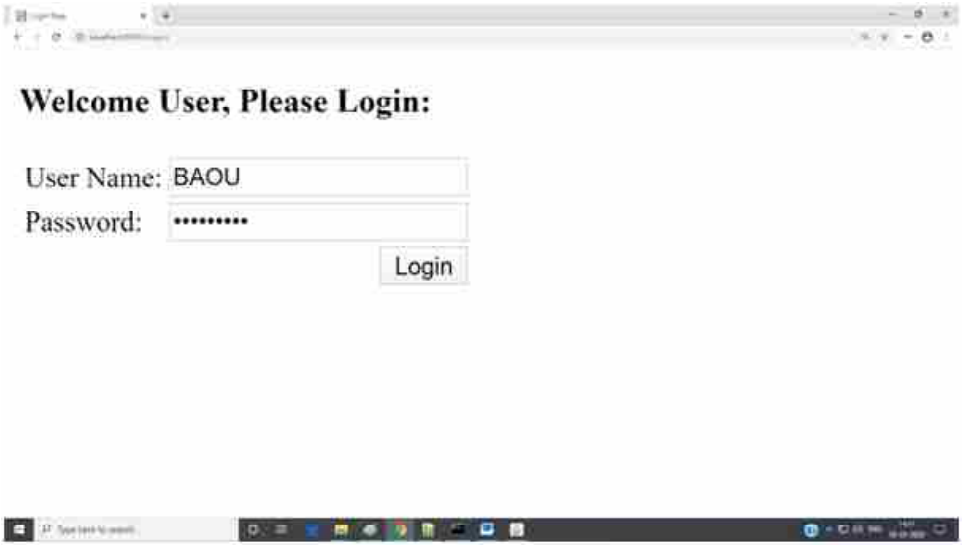
It will display following screen with list of applications :



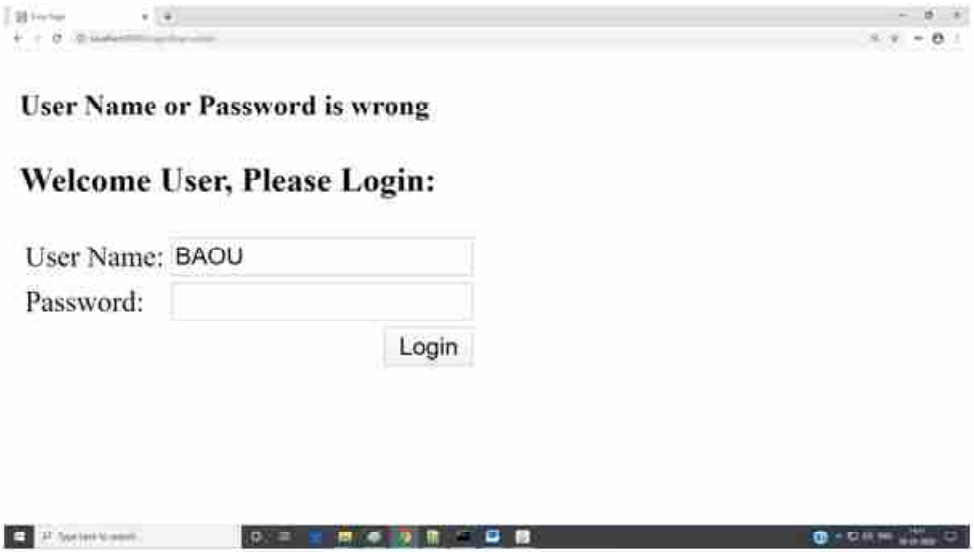
Now, Click on Login Application on the left pane. It will display index.jsp page as shown below.



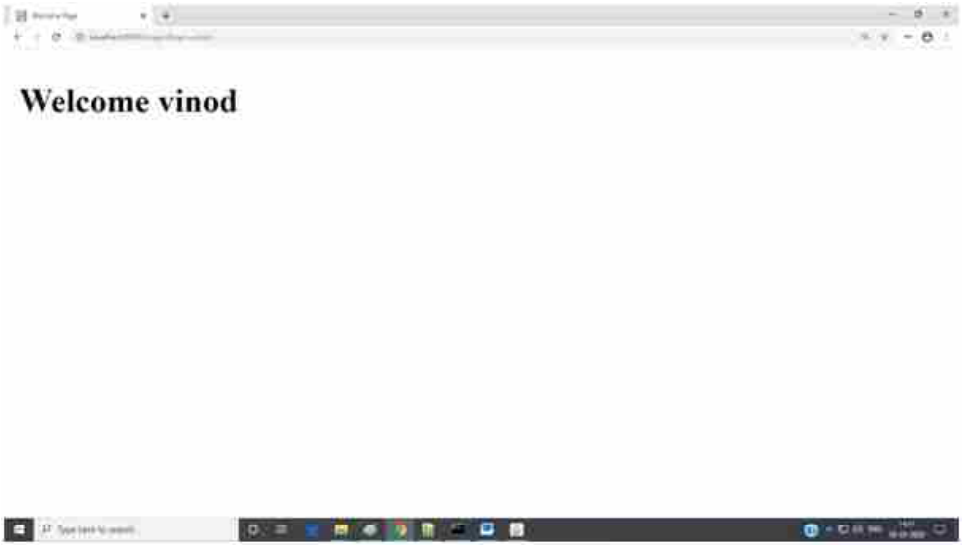
Now, First Provide invalid username, password and click on login button as shown below :



With wrong credentials it will redirect to index.jsp with proper message as shown below :



Now, Provide correct credentials and check the login page as shown below. It displays welcome message with user name.



Notice that the URL changes to the action in the address bar. So, this way we can build, deploy and test other Struts application.

❑ **Check Your Progress – 6 :**

1. Which of the following delegates the request handling to the RequestProcessor instance ?

   a. ActionServlet b. Action class

   c. Deployment descriptor d. None of the above

13.8 Let Us Sum Up :

While studying this unit, we have learnt that Struts is an application development framework that is designed for and used with the popular J2EE (Java 2, Enterprise Edition) platform.

The controller is responsible for intercepting and translating user input into actions to be performed by the model.

Controller components coordinate activities in the application. This may mean taking data from the user and updating a database through a Model component, or it may mean detecting an error condition with a back-end system and directing the user through special error processing.

Struts application is a normal web application that carries Struts libraries and configuration files. Such application can be created in similar manner as you create any other web application in IDE with the help of New Web Application wizard where extra step shows Struts libraries along with configuration files which can be added in an application.

13.9 Answer for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (e), **2.** (c)

❑ **Check Your Progress 2 :**

1. (b), **2.** (c)

❑ **Check Your Progress 3 :**

1. (c)

❑ **Check Your Progress 4 :**

1. (c)

❑ **Check Your Progress 5 :**

1. (a)

❑ **Check Your Progress 6 :**

1. (d)

13.10 Glossary :

1. **Code :** A number that uniquely identifies a catalog entry in the WebSphere Commerce system.
2. **Data bean :** A type of bean that is placed in a JSP file.

3. **POJO :** In struts 2, action class is POJO (Plain Old Java Object). POJO means we are not forced to implement any interface or extend any class.
4. **OGNL :** It stands for Object Graph Navigation Language.

13.11 Assignment :

1. Discuss various features of Struts.
2. Write short note on Struts Action Classes.
3. Write advantages and disadvantages of Struts 2.

13.12 Activities :

Collect some information on cconfiguring web.xml file for Struts.

13.13 Case Study :

Generalised the basic of Struts View Components.

13.14 Further Readings :

1. https://struts.apache.org/
2. https://www.javatpoint.com/struts-2-tutorial
3. https://www.tutorialspoint.com/struts_2/index.htm

Unit
14

JSP (JAVA SERVER PAGES)

UNIT STRUCTURE

14.0 Learning Objectives :

After learning this Unit, you will be:

- Define JSP Architecture
- Write JSP Syntax
- Define JSP Architecture
- Use JSP Implicit Objects
- Use JSTL functions

14.1 Introduction :

JSP is a server side technology which helps to create a webpage dynamically using java as the programming language. JSP is a specification from Sun Microsystems. It is an extension to Servlet API. It controls content or appearance of Web pages through application of servlets, which are small programs that are specified in Web page and run on Web server to alter Web page before it sent to user who requested it.

JSP enables developers to write HTML pages containing tags, inside which we can include powerful Java programs. Using JSP, we can easily separate Presentation and Business logic. Both the layers can easily interact over HTTP requests.

14.2 JSP Life Cycle :

JSP files are saved with .jsp extension which lets the server identify that this is a JSP page and needs to go through JSP life cycle stages. When the client makes a request to Server, it first goes to container. Then container checks whether the servlet class is older than jsp page (To check whether the JSP file is modified). If it is then container does the translation again (converts JSP to Servlet) otherwise it skips the translation phase (i.e. it doesn't do the translation to improve the performance as this phase takes time and to repeat this step every time is not time feasible).

Life Cycle of JSP page consists of :

1. When container receives request from client, it does translation only when servlet class is older than JSP page otherwise it skips this phase.
2. Then the container compiles the corresponding servlet program and loads the corresponding servlet class. After it instantiates the servlet class and Calls the jspInit() method to initialize the servlet instance (Jsp container will do this job only when the instance of servlet file is not running or if it is older than the jsp file.)
3. A new thread is then gets created, which invokes the_jspService() method, with a request (HttpServletRequest) and response (HttpServletRespnse) objects as parameters.
4. Invokes the jspDestroy() method to destroy the instance of the servlet class.

❑ **Check Your Progress – 1 :**

1. Which one is the correct order of phases in JSP life cycle ?

   a. Initialization, Cleanup, Compilation, Execution

   b. Initialization, Compilation, Cleanup, Execution

   c. Compilation, Initialization, Execution, Cleanup

   d. Cleanup, Compilation, Initialization, Execution

2. Which of the following step is taken by JSP container during Compilation phase ?

   a. Parsing the JSP. b. Turning the JSP into a servlet.

   c. Compiling the servlet. d. All of the above.

14.3 JSP Architecture :

Depending on the location of request processing, Servlet OR JSP carries two architectures which are:

Model 1 Architecture:

In this Model, JSP plays a key role and it is responsible for of processing the request made by client. Client (Web browser) makes a request, JSP then creates a bean object which then fulfills the request and pass the response to JSP. JSP then sends the response back to client. Unlike Model2 architecture in this Model, most of the processing is done by JSP itself.

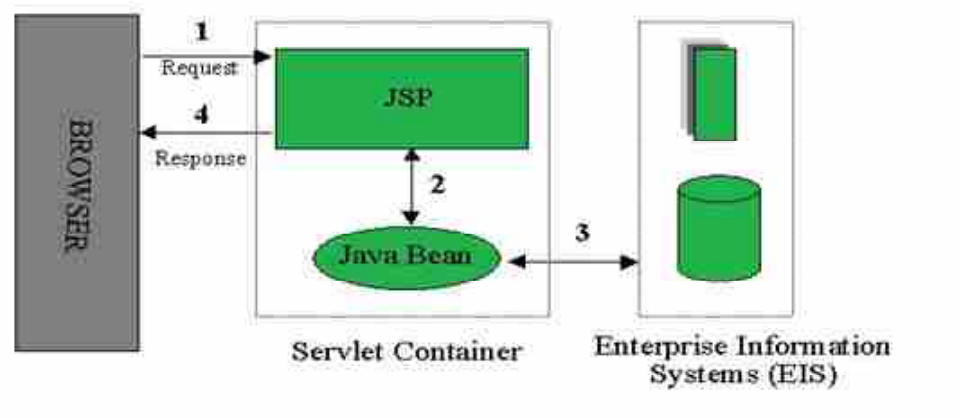


Fig 3.1 Architecture Model 1

Advantage :

- This model is Easy and Quick to develop web application.

Disadvantage :

- Decentralized navigation control: As every page contains the logic to determine the next page. If JSP page name is changed that is referred by other pages, we need to change it in all the pages that leads to the maintenance problem.
- Time consuming: We need to spend more time to develop custom tags in JSP. So that we don't need to use scriptlet tag.
- Hard to extend: It is better for small applications but not for large applications.

Model 2 Architecture :

The JSP Model 2 architecture is based on the popular MVC architecture. In this Model Servlet plays a major role and it is responsible for processing the client's(web browser) request. Presentation part (GUI part) will be handled by JSP and it performs this with the help of bean as shown in below figure. The servlet acts as controller and in charge of request processing. It creates the bean objects if required by the jsp page and calls the respective jsp page. The jsp handles the presentation part by using the bean object. In this Model, JSP doesn't do any processing, Servlet creates the bean Object and calls the JSP program as per the request made by client.

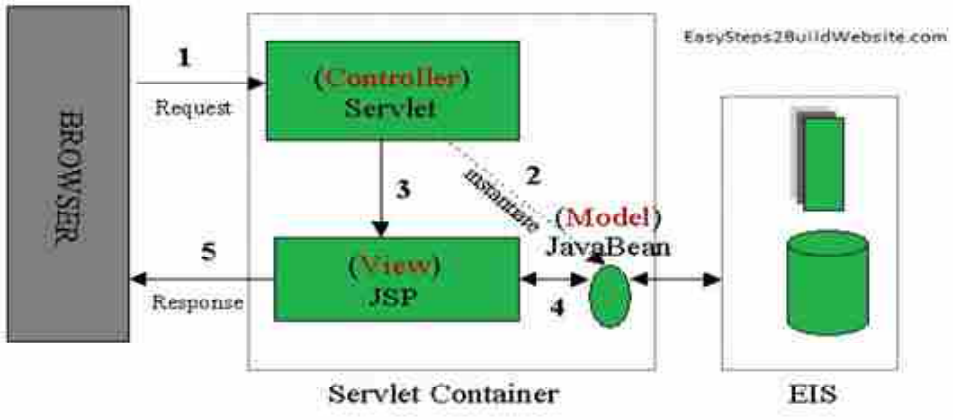


Fig 3.2 Architecture Model 2

Advantage :

- Here, the navigation control is centralized, only controller contains the logic to determine the next page.
- It is easy to maintain
- It is easy to extend
- It is easy to test

Disadvantage :

- We need to write the controller code self. Suppose, if we change the controller code, we need to recompile the class and redeploy the application.

❑ **Check Your Progress – 2 :**

1. For what JSP is used ?

   a. Server-side dynamic content generation

   b. Client Side language for validation

   c. None of the above

14.4 JSP Basic Building Block :

Any JSP page generally consists of following components:

1. Declaration

A declaration tag is a piece of Java code for declaring variables, methods and classes. If we declare a variable or method inside declaration tag it means that the declaration is made inside the servlet class but outside the service method.

We can declare a static member, an instance variable (can declare a number or string) and methods inside the declaration tag.

Syntax of declaration tag:

```
<%!  Declaration %>
```

E.g.

```
<%! String name="Dimple" %>
<%! int age=30; %>
```

In above code we have declared two variables inside declaration tag.

In below code we have declared a method named sum.

```
<%!
        int sum(int num1, int num2, int num3)
    {
            return num1+num2+num3;
    }
%>
```

2. Scriptlet

Scriptlets are nothing but java code enclosed within <% and %> tags. JSP container moves statements in _jspservice() method while generating servlet from jsp. For each request from the client, service method of the JSP gets invoked hence the code inside the Scriptlet executes for every request. A Scriptlet contains java code that is executed every time JSP is invoked.

Syntax of Scriptlet tag:

```
<% java code %>
```

In below code, we are taking Scriptlet tags which enclose java code.

```
<%      int num1=20;
        int num2=30;
        int sum = num1+num2;
        out.println("Scriplet Number is " +sum);
%>
```

3. Expression

Expression tag evaluates the expression placed in it, converts the result into String and send the result back to the client through response object. Generally it writes the result to the client i.e. browser.

Syntax of expression tag:

```
<%= expression %>
```

E.g.

```
<%
        int a=30;
        int b=40;
        int c=50;
%>
        <%= a+b+c %>
```

4. Comment

Comments are the one when JSP container wants to ignore certain texts and statements. When we want to hide certain content, then we can add that to the comments section.

Syntax:

```
<% -- JSP Comments %>
```

❑ **Check Your Progress – 3 :**

1. Which of the scripting of JSP not putting content into service method of the converted servlet ?

   a. Declarations b. Scriptlets

   c. Expressions d. None of the above

14.5 JSP Implicit Objects :

JSP provide access to some implicit object which represent some commonly used objects for servlets that JSP page developers might need to use. For example we can retrieve HTML form parameter data by using request variable, which represent the HttpServletRequest object. There are total 9 implicit objects available in JSP. Following table shows the JSP implicit object:

Implicit Object	Description
request	The HttpServletRequest object associated with the request. It belongs to javax.servlet.http.HttpServlet Request. **Example:** Request User-Agent: <%=request.getHeader("User-Agent") %>
response	The HttpServletRequest object associated with the response that is sent back to the browser. It belongs to javax.servlet.http.HttpServletResponse. **Example:** <% response.setContentType("text/html"); %> <% response.addCookie(new Cookie("Name", "BAOU")); %>

Out	The JspWriter object associated with the output stream of the response. Belongs to package javax.servlet.jsp. jspwriter.
	Example:
	<% int num1=30; int num2=50;
	out.println("num1 is: " +num1);
	out.println("num2 is: "+num2);
	%>

session	The HttpSession object associated with the session for the given user of request. It belongs to javax.servlet.http.HttpSession. **Example:** <% session.setAttribute("user", "BAOU"); %>
application	This is used for getting application-wide initialization parameters and to maintain useful data across whole JSP application. The ServletContext object for the web application. Belongs to package Javax.servlet. ServletContext. **Example:** <% application.getContextPath(); %> User context param value :<%=application.getInitParameter("UserName") %>
Config	The ServletConfig object associated with the servlet for current JSP page. This is a Servlet configuration object and mainly used for accessing getting configuration information such as servlet context, servlet name, configuration parameters etc. Belongs to package Javax.servlet.ServletConfig.

	Example: <% String servletName = config.getServletName(); out.println("Servlet Name is: " + servletName); %>
pageContext	The PageContext object that encapsulates the enviroment of a single request for this current JSP page. It is used for accessing page, request, application and session attributes. Belongs to package java.servlet.jsp.PageContext. **Example:** <% pageContext.setAttribute("userName", "BAOU", pageContext.PAGE_SCOPE); String name = (String) pageContext.getAttribute ("userName"); out.println("User name is: " + name); %>
Page	The page variable is equivalent to this variable of Java programming language. **Example:** <% String pageName = page.toString(); out.println("Page Name is: " + pageName);%>
exception	The exception object represents the Throwable object that was thrown by some other JSP page. **Example:** <% int[] baou={1,2,3,4}; out.println(baou[5]); %> <%= exception %>

It has an array of numbers, i.e., baou with four elements. We are trying to print the fifth element of the array from baou, which is not declared in the array list. So it is used to get exception object of the jsp. Here, We will get ArrayIndexOfBoundsException in the array.

❑ **Check Your Progress – 4 :**

1. How many jsp implicit objects are there and these objects are created by the web container that are available to all the jsp pages?

   a. 8 b. 9 c. 10 d. 7

14.6 Standard Actions :

JSP actions use the construct in XML syntax to control the behavior of the servlet engine. We can dynamically insert a file, reuse the beans components, forward user to another page, etc. through JSP Actions like include and forward. Unlike directives, actions are re-evaluated each time the page is accessed. These tags are used to remove or eliminate scriptlet code from our JSP page because scriplet code are technically not recommended nowadays. It's considered to be bad practice to put java code directly inside your JSP page.

Standard tags begin with the jsp: prefix. There are many JSP Standard Action tags which are used to perform some specific task.

Syntax :

<jsp:action_name attribute="value" />

The following are some JSP Standard Action Tags available :

Action Tag	Description
jsp:forward	It forwards the request to another page. Syntax of <jsp:forward> : <jsp:forward page="URL of the another static, JSP OR Servlet page" />
jsp:useBean	It instantiates a JavaBean. This action is useful when we want to use Beans in a JSP page, through this tag we can easily invoke a bean. Syntax of <jsp:useBean>: <jsp: useBean id="unique_name_of_bean" class="package_name.class_name" /> Once Bean class is instantiated using above statement, we have to use jsp:setProperty and jsp:getProperty actions to use the bean's parameters.
jsp:getProperty	It is used to retrieve or fetch the value of Bean's property. syntax of <jsp:getProperty>: <jsp: useBean id="unique_name_of_bean" class="package_name.class_name" /> .... <jsp:getProperty name="unique_name_of_bean" property="property_name" />
jsp:setProperty	It store data in property of any JavaBeans instance. This action tag is used to set the property of a Bean, while using this action tag, we may need to specify the Bean's unique name (it is nothing but the id value of useBean action tag). syntax of <jsp:setProperty>: <jsp: useBean id="unique_name_of_bean" class="package_name.class_name" /> .... <jsp:setProperty name="unique_name_of_bean" property="property_name" />
jsp:include	It includes the runtime response of a JSP page into the current page. In <jsp:include> the file is being included during request processing. Syntax of <jsp:include> : <jsp:include page="page URL" flush="Boolean Value" />

jsp:plugin	It generates client browser-specific construct that makes an OBJECT or EMBED tag for the Java Applets. It is used to introduce Java components into jsp, i.e., the java components can be either an applet or bean. It detects the browser and adds <object> or <embed> tags into the file Syntax: <jsp:plugin type="applet/bean" code="objectcode" codebase="objectcodebase">
jsp:fallback	It supplies alternate text if java plugin is unavailable on the client. You can print a message using this, if the included jsp plugin is not loaded.
jsp:element	Defines XML elements dynamically
jsp:attribute	It defines dynamically defined XML element's attribute. This tag is used to define the XML dynamically i.e. the elements can be generated during request time than compilation time It actually defines the attribute of XML which will be generated dynamically. Syntax: <jsp:attribute></jsp:attribute>
jsp:body	Used within standard or custom tags to supply the tag body. This tag is used to define the XML dynamically i.e., the elements can generate during request time than compilation time. It actually defines the XML, which is generated dynamically element body. Syntax: <jsp:body></jsp:body>
jsp:param	Adds parameters to the request object. Syntax of <jsp:param>: <jsp: param name="param_name" value="value_ of_parameter" />
jsp:text	It is used to template text in JSP pages. Its body does not contain any other elements, and it contains only text and EL expressions. Syntax: <jsp:text>Welcome Message</jsp:text>

❑ **Check Your Progress – 5 :**

1. Which tag should be used to pass information from JSP to included JSP ?

   a. Using <%jsp:page> tag b. Using <%jsp:param> tag

   c. Using <%jsp:import> tag d. Using <%jsp:useBean> tag

2. Which is mandatory in <jsp:useBean /> tag ?

   a. id, class b. id, type

   c. type, property d. type,id

14.7 JSP Tag Libraries :

JSTL stands for Java server pages standard tag library, and it is a collection of custom JSP tag libraries that provide common web development functionality.

The JSTL contains several tags that can remove scriptlet code from a JSP page by providing some ready to use, already implemented common functionalities.

JSTL is divided into 5 groups :

1. **JSTL Core :** JSTL Core provides several core tags such as if, forEach, import, out etc to support some basic scripting task. Url to include JSTL Core Tag inside JSP page is :

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

2. **JSTL Formatting :** JSTL Formatting library provides tags to format text, date, number for Internationalised web sites. Url to include JSTL Formatting Tags inside JSP page is:

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

3. **JSTL sql :** JSTL SQL library provides support for Relational Database Connection and tags to perform operations like insert, delete, update, select etc on SQL databases. Url to include JSTL SQL Tag inside JSP page is:

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

4. **JSTL XML :** JSTL XML library provides support for XML processing. It provides flow control, transformation features etc. Url to include JSTL XML Tag inside JSP page is:

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

5. **JSTL functions :** JSTL functions library provides support for string manipulation. Url to include JSTL Function Tag inside JSP page is:

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

JSTL Core

The core tags are most frequently used tags in JSP. They provide support for

- Iteration
- Conditional logic
- Catch exception
- url forward
- Redirect, etc.

To use core tags we need to define tag library first and below is the syntax to include a tag library.

Syntax :

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

Example :

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
   <head>
      <title>Tag Example</title>
   </head>
   <body>
         <c:if test="${param.name == 'baou'}">
               <p>Welcome to ${param.name} </p>
         </c:if>
         <c:out value="${param.name}" default="baou" />
         <c:forEach var="message" items="${errorMsgs}" >
               <li>${message}</li>
         </c:forEach>
   </body>
</html>
```

❑ **Check Your Progress – 6 :**

1. What JSTL stands for ?
   a. JavaServer Pages Standard Tag Library
   b. JSP Tag Library
   c. Java Standard Tag Library
   d. None of the above.
2. Which of the following is an advantage of the statement, Separation of business logic from JSP ?
   a. Custom Tags in JSP b. JSP Standard Tag Library
   c. All the above d. None of the above

14.8 Let Us Sum Up :

In this unit we have learnt that JSP is a server side technology which helps to create a webpage dynamically using java as the programming language.

JSP is Java Server Pages while ASP is Active Server Pages which are two commonly used server side scripting languages used today in web development.

To explain the syntax of JSP, we have to first start with the Scriptlet. A scriptlet contain any number of JAVA language statements, variable or method declarations, or expressions which is valid in page scripting language.

JSP provides Standard Action tags which are applied inside JSP pages. Such tags are used to remove or eliminate scriptlet code from JSP page as scriplet code are technically not suggested these days.

14.9 Answer for Check Your Progress :

- **Check Your Progress 1 :**

  1. (c), **2.** (c)

- **Check Your Progress 2 :**

  1. (a)

- **Check Your Progress 3 :**

  1. (c)

- **Check Your Progress 4 :**

  1. (b)

- **Check Your Progress 5 :**

  1. (b), **2.** (a)

- **Check Your Progress 6 :**

  1. (a), **2.** (a)

14.10 Glossary :

1. JSP : A server technology used to create webpage in java.

2. Presentation : A Graphical View presented to the user

14.11 Assignment :

1. Explain the JSP Architecture ?
2. List the benefits of JSP.
3. Explain JSP page execution in detail.
4. Explain different JSP Directives.

14.12 Activities :

Study and implement JSP Implicit Objects.

14.13 Case Study :

Study and implement the types of JSP Tag Libraries.

14.14 Further Readings :

1. https://www.tutorialspoint.com/jsp/jsp_overview.htm
2. https://www.guru99.com/jsp-tutorial.html
3. https://beginnersbook.com/2013/05/jsp-tutorial-introduction/

BLOCK SUMMARY :

In this block, students have learnt and understand about the basic of Basic of JSP Syntax and Implicit Objects. The block gives an idea on the study and concept of basic Struts Action Classes and Model Components. The students have be well explained on the concepts of compiling and running Servlets in Apache Tomcat 4.0.

The block detailed about the basic idea of Threading Issues techniques. The concept related to Struts Action Classes and Model Components will also be explained to the students. The student will be demonstrated practically about Architecture of the Servlet Package.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. What is Struts Action Classes ?
2. Explain the Threading Issues?
3. Write note on Struts Controller Components ?
4. Write short note on JSP Syntax and Implicit Objects ?

❖ **Long Questions :**

1. Write short notes on Architecture of the Servlet Package ?
2. Write short note on compiling and running Servlets in Apache Tomcat 4.0 ?
3. Explain various JSTL formatting function with example.
4. Write note on configuring web.xml file for Struts ?

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	12	13	14
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	_____
Language and Style	☐	☐	☐	☐	_____
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	_____
Conceptual Clarity	☐	☐	☐	☐	_____
Check your progress Quest	☐	☐	☐	☐	_____
Feed back to CYP Question	☐	☐	☐	☐	_____

3. Any other Comments

.....

.....

.....

.....

.....

.....

.....

.....