		L	T	P	C
17BECS5E07	**SERVLETS AND JSP**	**3**	**0**	**0**	**3**

COURSE OBJECTIVES:

- Write servlets using the Java programming language (Java servlets)
- Understand and manage HTTP sessions in a web application
- Create servlet filters and listeners
- Write pages created with Java Server Pages technology (JSP pages)
- Create easy-to-maintain JSP pages using the Expression Language and the JSP Standard Tag Library (JSTL)
- Use integrated development environments (IDEs) and application servers for Java EE development and deployment

LEARNING OUTCOMES:

- Construct and deploy small-to-medium scale web applications found in intranet and low-volume commercial sites by using JavaServer Page (JSP page) technology and servlets.
- Apply Model-View-Controller (MVC) architecture to projects in EE environments.
- Create servlet filters and listeners.
- Understand and manage HTTP sessions in a web application.
- Create easy-to-maintain JSP pages using Expression Language and the JSP Standard Tag Library (JSTL).
- Analyze, design, develop and deploy web applications with Java EE 6 SDK and the application server Oracle WebLogic Server

UNIT I SERVLETS (9)

Web Application - Java Servlets - Servlet Lifecycle - Servlet Context - Session management - Building the first Servlet - Deploying the Servlet

UNIT II INTRODUCTION TO JSP (9)

Introduction to Java Server Pages - Features of JSP - Basic HTML Tags - JSP Tag library - JSP Page Life cycle - Developing a Simple Java server Page - - JSP Processing Model - Comments and Character Coding - MVC architecture - 3-tier architecture - Advantages of JSP over competing technologies

UNIT III JSP SCRIPTING ELEMENTS AND DIRECTIVES (9)

Forms of Scripting Elements - Predefined Variables - Examples using Scripting Elements - JSP Directives - JSP Page Directive - JSP Include Directive

UNIT IV JSP ACTIONS AND CUSTOM TAGS (9)

JSP Actions - include Action - forward Action - plugin Action - Java Beans - Bean Related – Actions - Custom Tag - Types of Tags - Creating Custom Tags

UNIT V ADVANCE CUSTOM TAGS AND JSTL (9)

Introduction - Using Simple Tag - Using tag files - JSP Standard Tag Library – purpose JSTL - Using Expression Language - Using JSTL

Total Hours: 45

TEXT BOOKS:

1. Mahesh P. Matha, "JSP and Servlets: A Comprehensive Study", Prentice-Hall of India Pvt.Ltd, 2013.
2. Joel Murach and Michael Urban," Murachs Java Servlets & JSP ", 3rd Edition, 2014.
3.Giulio Zambon" Beginning JSP, JSF and Tomcat: Java Web Development", Apress Kindle edition, 2012.

REFERENCES:

1. Santosh Kumar K , "Jdbc, Servlets, And Jsp Black Book", Dreamtech Press , New edition 2008.
2. Panduranga, S.N., Goyal, "Beginning Jsp 2", Springer/A Press ,Edition1,2004.
3. Phil Hanna, "The Complete reference JSP 2.0", Tata McGraw-Hill Education, 2003.

WEBSITES:

1. www.jsptut.com/
2. www.**tutorial**spoint.com/**jsp**/
3. www.javatpoint.com/**jsp-tutorial**

Lesson Plan

S.No.	Unit No.	Date (s)	Topics to be covered	No. of classes
1			Introduction to web technologies and their applications	1
2			Introduction to HTML and create to List, Tables	2
3			Create Images and forms	2
4	**I**		Frames	1
5			Cascading Style sheets	2
6			Examples for CSS HTML tags	2
7			Tutorial Class	1
8			Assignment Test – 1	1
9			Introduction to Java Script	1
10			Objects in Java Script	2
11	**II**		Dynamic HTML with Java Script	2
12			Programs on Java Script	2
13			Revise University Question papers	1

14			Assignment Test – 2	1
15			Introduction to XML	1
16			Document type definition	1
17			XML Schemas	1
18			Document Object model	1
19			Presenting XML	1
20	**III**		Using XML Processors: DOM and SAX	1
21			Examples for XML Programs	1
22			Revise University Question papers	1
23			Tutorial Class	1
24			Assignment Test – 3	1
25			Introduction to Java Beans, Advantages of Java Beans	1
26			BDK Introspection, Using Bound properties	1
27			Bean Info Interface, Constrained properties	1
28			Persistence, Customizes, Java Beans API	1

29	IV		Introduction to EJB's	1
30			Difference B/W Java Beans & EJB	1
31			EJB Client/Server Model & Process	1
32			Examples on EJB Applications	1
33			Revise University Question papers	1
34			Tutorial Class	
35			Assignment Test – 4	1
36	V		Tomcat web server, Introduction to Serve lets	1
37			Lifecycle of a Servlets, JSDK, The Servlets API	1
38			The javax.servlet Package, Reading Servlets parameters, Reading Initialization parameters	2
39			The javax.servlet HTTP package, Handling Http Request & Responses	1
40			Using Cookies-Session Tracking, Security Issues	1
41			Revise University Question papers	1

42			Tutorial Class	1
43			Assignment Test – 5	1
44			Introduction to JSP and examples of JSP applications	1
45			The Problem with Servlets. The Anatomy of a JSP Page	1
46			JSP processing	1
47	**VI**		JSP Application Design with MVC Setting Up and JSP Environment	1
48			Installing the Java Software Development Kit, Tomcat Server & Testing Tomcat	1
49			Revise University Question papers	1
50			Assignment Test – 6	1
51			Generating Dynamic Content, Using Scripting Elements	1
52			Implicit JSP Objects	1

53			Conditional Processing – Displaying Values Using an Expression to Set an Attribute	1
54	**VII**		Declaring Variables and Methods Error Handling and Debugging	1
55			Sharing Data Between JSP pages	1
56			Requests, and Users Passing Control and Date between Pages	1
57			Memory Usage Considerations	1
58			Revise University Question papers	1
59			Tutorial Class	1
60			Assignment Test – 7	1
61	**VIII**		Database Programming using JDBC	1
62			Studying Javax.sql.* package	2
63			Accessing a Database from a jsp page	1

64			Specific Database Actions	1
65			Deploying JAVA Beans in a JSP Page	1
66			Introduction to struts framework	1
67			Revise University Question papers	1
68			Tutorial Class	1
69			Assignment Test – 8	1

LECTURE NOTES

UNIT – 1

OVER VIEW:

Unit-1 demonstrates how to create static web pages using basic HTML tags and presentation of content using CSS. This unit focuses on how to create tables, forms, and lists to display the content in web pages.

CONTENTS:

1. Basic HTML Tags
    a. List
    b. Tables
    c. Images

d. Forms

e. frames

2. Cascading Style Sheets

a. Three mechanisms by which we can apply styles

b. Forms of CSS

HTML stands for Hypertext Markup Language. It is used to display the document in the web browsers. HTML pages can be developed to be simple text or to be complex multimedia program containing sound, moving images and java applets. HTML is considered to be the global publishing format for Internet. It is not a programming language. HTML was developed by Tim Berners-Lee.

HTML standards are created by a group of interested organizations called W3C (world wide web consortium). In HTML formatting is specified by using tags. A tag is a format name surrounded by angle brackets. End tags which switch a format off also contain a forward slash.

Basic HTML tags

1. Body tag :

Body tag contain some attributes such as bgcolor, background etc. bgcolor is

used for background color, which takes background color name or hexadecimal

number and #FFFFFF and background attribute will take the path of the image

which you can place as the background image in the browser.
<body bgcolor="#F2F3F4" background= "c:\amer\imag1.gif">

2. Paragraph tag:

Most text is part of a paragraph of information. Each paragraph is aligned to the

left, right or center of the page by using an attribute called as align.

<p align="left" | "right" | "center">

3. Heading tag:

HTML is having six levels of heading that are commonly used. The largest heading tag is <h1> . The different levels of heading tag besides <h1> are <h2>,

<h3>, <h4>, <h5> and <h6>. These heading tags also contain attribute called

as

align.

<h1 align="left" | "right" | "center"> . . . . <h2>

4. hr tag:

This tag places a horizontal line across the system. These lines are used to break

the page. This tag also contains attribute i.e., width which draws the horizontal

line with the screen size of the browser. This tag does not require an end tag.

<hr width=”50%”>.

5. base font:

This specify format for the basic text but not the headings.

<basefont size=”10”>

6. font tag:

This sets font size, color and relative values for a particular text.

<font size=”10” color=”#f1f2f3”>

7. bold tag:

This tag is used for implement bold effect on the
text …….

8. Italic tag:

This implements italic effects on the text.

…….

9. strong tag:

This tag is used to always emphasized the text

…….

10. tt tag:

This tag is used to give typewriting effect on the text

<tt>……..</tt>

11. sub and sup tag:

These tags are used for subscript and superscript effects on the text.

_{…….}

^{……..}

12. Break tag:

This tag is used to the break the line and start from the next line.

13. &amp &lt &gt   &quot

These are character escape sequence which are required if you want to display

characters that HTML uses as control sequences.
Example: < can be represented as &lt.

14. Anchor tag:

This tag is used to link two HTML pages, this is represented by <a>

 some text

href is an attribute which is used for giving the path of a file which you want to

link.

Lists:

One of the most effective ways of structuring a web site is to use lists. Lists provides

straight forward index in the web site. HTML provides three types of list i.e.,

1. **unordered list,**
2. **ordered list and**
3. **definition list.**

Lists can be easily embedded easily in another list to provide a complex but

readable structures. The different tags used in lists are explained below.

 …..

The ordered(numbered) and unordered(bulleted) lists are each made up of sets of list items. This tag is used to write list items

1. **unordered list,**

<ul type=”disc” | “square” | ”circle” > …..

This tag is used for basic unordered list which uses a bullet in front of each tag, every

thing between the tag is encapsulated within <li> tags.

2. **ordered list,**

<ol type=”1” | ”a” | “I” start=”n”>…..

This tag is used for unordered list which uses a number in front of each list item or it uses any element which is mentioned in the type attribute of the <ol> tag, start attribute is used for indicating the starting number of the list.

3. **definition list.**

<dl>….. </dl>

This tag is used for the third category i.e., definition list, where numbers or bullet is not used in front of the list item, instead it uses definition for the items.

<dt>…..</dt>

This is a sub tag of the <dl> tag called as definition term, which is used for marking the items whose definition is provided in the next data definition.

<dd> ….</dd>

This is a sub tag of the <dd> tag, definition of the terms are enclosed within these tags. The definition may include any text or block.

Tables:

Table is one of the most useful HTML constructs. Tables are find all over the web application. The main use of table is that they are used to structure the pieces of

information and to structure the whole web page. Below are some of the tags used in

table.

Links

The HTML code for a link is simple. It looks like this:

Link text

Frames:

Frames provide a pleasing interface which makes your web site easy to navigate. The frameset contains a set of references to HTML files, each of which is displayed inside a separate frame.

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The frameset element holds two or more frame elements. Each frame element holds a separate document.

Forms:

Forms are the best way of adding interactivity of element in a web page. They are usually used to let the user to send information back to the server but can also be used to simplify navigation on complex web sites. The tags that use to implement forms are as follows.

<form action=”URL” method = “post” | “get”>…….</form>

Cascading Style sheets:

One of the most important aspects of HTML is the capability to separate presentation and content. A style is simply a set of formatting instructions that can be applied to a piece of text.

There are three mechanisms by which we can apply styles to our HTML documents.

· **Inline Style Sheet:**

Style can be defined within the basic HTML tag.

· Internal Style Sheet:

Style can be defined in the <head>
tag **· External Style Sheet:**

Styles can be defined in external files called stylesheets which can then be used

in

any document by including the stylesheet via a URL.

UNIT – II

Overview :

Unit-II demonstrates on client side validations using JavaScript. It focuses on how to handle events, exceptions, etc. This unit also focuses on DHTML concepts.

CONTENTS:

1) JavaScript concepts
2) Objects in java scripts
3) DHTML with JavaScript

JavaScript (also called JScript) is a scripting language with the primary aim of giving life to our web pages.It is very powerful, flexible, and easy to learn.

Features

- Imperative and structured
- Dynamic
    - dynamic typing
    - object based
    - run-time evaluation
- Functional
    - first-class functions
    - nested functions
    - closures
- Prototype-based
- Miscellaneous
- Vendor-specific extensions

There are three ways by which we can place Javascript for use in a web page.

1. Inside the head section.

2. Within the body section.

3. In an external file.

The HTML <script> tag is used to insert a Java Script into an HTML page.

Events

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by Java Script.

Every element on a dynamic or static web page has certain events which can trigger Java Script functions. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on that button. We define the events in the HTML tags.

Examples of events:

A mouse click

A web page or an image loading

Mousing over a hot spot on the web page

Selecting an input box in an HTML form

Submitting an HTML form

A keystroke

The following table lists the events recognized by JavaScript:

Events are normally used in combination with functions, and the function can not be executed before the event occurs.

JavaScript - Catching Errors

There are two ways of catching errors in a Webpage

Try...Catch Statement

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

```
try
{
//Run some code here
}

catch(err)
{
//Handle errors here
}
```

The onerror Event

The onerror event can be explained soon, but first you will learn how to use the throw statements to create an exception. The throw statements can be used together with the try...catch statement.

The Date object

The Date class is used to store and retrieve dates in JavaScript.

Array

The Array object is used to holding a set of data or values in a single variable name.

var urArray=new Array()

DOM (document object model)

A DOM (document object model) is an application programming interface (API) for representing a document (such as an HTML document) and accessing and manipulating the various elements (such as HTML tags and strings of text) that make up that document. Java Script-enabled web browsers have always defined a document object model; a web-browser DOM may specify, for example: that the forms in an HTML document are accessible through the forms[] array of the Document object.

form validation

Form validation is the process of checking that a form has been filled in correctly or not before it is processed.

There are two methods for the form validation.

1: Client-Side validation

In Java Script Client-side form validation is an important part of a web site where data needs to be collected from the user. Users are innately ignorant, and will mess up data entry in a web form if given the chance. It is the job of the web programmer, then, to make sure his pages which use forms include client-side form validation using JavaScript.

2: Server-side validation

In Java Script the server also benefits from client-side validation since it saves a number of round-trips between the visitor and the server owing to typos and easily spotted mistakes. This advantage does not alleviate the neccessity of doing server-side validation.

UNIT - III

Overview :

This unit focuses on creating XML documents that are designed to carry data. XML is all about **describing** information. XML was designed to transport and store data. We can create web documents from XML using XSLT to transform our documents into HTML. We can then send

our XML to an XSLT processor on the web server and serve that result to the web browser. This makes our documentation available in whatever format we need it to be in.

Contents :

Introduction to XML

DTD

XML Schema

XSLT

DOM

SAX

Introduction to XML

XML describes and focuses on the data while HTML only displays and focuses on how data looks. HTML is all about displaying information but XML is all about describing information. The tags used to mark up HTML documents and the structure of HTML documents are predefined. The author of HTML documents can only use tags that are defined in the HTML standard.

In HTML some elements can be improperly nested within each other like this:

This text is bold and italic

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

An XML document is composed of

1. Declarations (prolog, dtd reference)
2. Elements

3. Comments

4. Entities (predefined, custom defined, character entities)

The XML declaration: Always the first line in the xml document:

The XML declaration should always be included. It defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set.

<?xml version="1.0" encoding="ISO-8859-1"?>

Root Element: The next line defines the first element of the document . It is called as the root element

<E-mail>

Child Elements: The next 4 lines describe the four child elements of the root (To, From, Subject and Body).

And finally the last line defines the end of the root element .

</E-mail>

</E-mail>

syntax-rules

- All XML elements must have a closing tag
- XML tags are case sensitive
- XML Elements Must be Properly Nested
- XML Documents Must Have a Root Element
- Always Quote the XML Attribute Values
- With XML, White Space is Preserved
- Comments in XML <!-- This is a comment -->
- XML Elements have Relationships
    - Elements in a xml document are related as parents and children.

XML elements must follow these naming conventions:

Names must not start with a number or punctuation character but it can contain letters, numbers, and other characters without spaces. Names must not start with the letters xml (or XML, or Xml, etc)

XML Attributes

- XML elements can have attributes in the start tag, just like HTML.
- Attributes are used to provide additional information about elements.
- Attribute values must always be enclosed in quotes. Use either single or double quotes eg. <color="red"> or <color='red'>
- If the attribute value itself contains double quotes it is necessary to use single quotes, like in this example: <name='Rose "India" Net'>

- **:** If the attribute value itself contains single quotes it is necessary to use double quotes, like in this example: <name="Rose 'India' Net">

DTD(Document Type Definition**)**

A Document Type Definition (DTD) defines the legal building blocks of an XML document.It defines the document structure with a list of legal elements and attributes. A DTD can be defined inside a XML document, or a external reference can be declared .

Internal DTD

If the DTD is defined inside the XML document, it should be wrapped in a DOCTYPE definition with the following syntax:

<!DOCTYPE root-element [element-

declarations]> **External DTD**

If the DTD is defined in an external file, it should be wrapped in a DOCTYPE definition with the following syntax:

<!DOCTYPE root-element SYSTEM

"filename"> **Importance of a DTD**

- With a DTD, a XML file carries a description of its own format.
- With a DTD, independent groups of people can agree to use a standard DTD for interchanging data.
- User application can use a standard DTD to verify that the data he receives from the outside world is valid.
- User can also use a DTD to verify his own data.

Building blocks of XML DTD Documents:

- Elements
- Attributes
- Entities
- PCDATA
- CDATA

PCDATA

- PCDATA means **parsed character data.** It can be thought as the character data (text) found between the start tag and the end tag of a XML element.
- **PCDATA is a text to be parsed by a parser**. **The text is checked by the parser for entities and markup**.
- Tags inside the text will be treated as markup and entities will be expanded. However, parsed character data should not contain any **&, <,** or **>** characters. These should be represented by the **&amp** , **&lt,** and **&gt** entities, respectively.

CDATA:

- **CDATA is character data that will NOT be parsed by a parser**. Tags inside the text will NOT be treated as markup and entities will not be expanded.

DTD-Elements: Elements are the **main constituent components** of both XML documents.

Elements can contain text, other elements, or be empty.

Syntax:

<!ELEMENT element-name category>

or

<!ELEMENT element-name (element-content)>

EX: **Elements with Parsed Character Data**

<!ELEMENT To (#PCDATA)>

<!ELEMENT From (#PCDATA)>

Elements with Children (sequences)

Elements with one or more children are declared with the name of the children elements inside the parentheses as :

<!ELEMENT element-name (child1)>

or

<!ELEMENT element-name (child1,child2,...)>

EX:

<!ELEMENT E-mail (To,From,Subject,Body)>

When children are declared in a sequence separated by commas, the children must appear in the same sequence in the document.

In a full declaration, the children must also be declared.

Children can have children.

Tag qualifiers

* Indicates zero or more occurrence.

<!ELEMENT color (Fill-Red*)>

? Indicates Zero or one time occurrence.

<!ELEMENT color (Fill-Red?)>

+ Indicates one or more occurrence

<!ELEMENT color (Fill-Red+)>

( ) Indicates a group of expressions to be matched together. EX:<!ELEMENT E-mail(#PCDATA|To|From|Subject|Body)*>

| Indicates an option.

<!ELEMENT E-mail (To,From,Subject,(Message|Body))>

Special tag Values in DTD

- **Tag definition can have following instead of sub-tags:**
- **ANY**
  - Indicates that the tag can contain any other defined element or PCDATA.
  - Usually used for the root element.
  - Elements can occur in any order in such a document.
  - Not recommended to be used.
- **EMPTY**
  - It says that the element contains no contents (and consequently no corresponding end-tag)

- Ex: to allow a tag flag used as <flag/> in a xml file the DTD entry should be
  <!ELEMENT flag EMPTY>

DTD-Attributes:

Attributes provide **extra information about elements**.

In a DTD, attributes are declared with an ATTLIST declaration.

Declaring Attributes

The ATTLIST declaration defines the element having a attribute with attribute name , attribute type , and attribute default value. An attribute declaration has the following syntax:

<!ATTLIST element-name attribute-name attribute-type default-value>

DTD example:

<!ATTLIST reciept type CDATA "check">

XML example:

<reciept type="check" />

DTD-Entities

Entities are variables used to define shortcuts to standard text or special characters. Entity references are references to entities Entities can be declared internally or externally.

Internal Entity Declaration

Syntax:

<!ENTITY entity-name "entity-value">

XML Schema

XML Schemas are more powerful than DTDs.

XML Schema is a W3C Standard. It is an XML-based alternative to DTDs.

It describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

We think that very soon XML Schemas will be used in most Web applications as a replacement for DTDs. Here are some reasons:

- XML Schemas are extensible to future additions
- XML Schemas are richer and more powerful than DTDs
- XML Schemas are written in XML, supports data types and namespaces.

What is an XML Schema?

- XML Schema is used to define the legal building blocks of an XML document, just like a DTD.

- An XML Schema defines user-defined integrants like elements, sub-elements and attributes needed in a xml document.
- It defines the data types for elements and attributes along with the occurrence order .
- It defines whether an element is empty or can include text.
- It also defines default and fixed values for elements and attributes

Features of XML Schemas :

XML Schemas Support Data Types

One of the greatest strengths of XML Schemas is its support for data types. With support for data types:

- It is easier to describe allowable document content
- It is easier to validate the correctness of data
- It is easier to work with data from a database
- It is easier to define data facets (restrictions on data)
- It is easier to define data patterns (data formats)
- 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. Simple XML editors are used to edit the Schema files. Even the same XML parsers can be used to parse the Schema files.

XML Schemas are Extensible

XML Schemas are extensible, because they are written in XML.So a user can reuse a Schema in other Schemas and can also refer multiple schemas in the same document. He can also create his own data types derived from the standard types

XML Schemas Secure Reliable Data Communication

When sending data from a sender to a receiver, it is essential that both parts have the same "expectations" about the content. With XML Schemas, the sender can describe the data in a way that the receiver will understand. A date like: "03-11-2004" will, in some countries, be interpreted as 3.November and in other countries as 11.March.However, an XML element with a data type like this: <datetype="date">2004-03-11</date> ensures a mutual understanding of the content, because the XML data type "date" requires the format "YYYY-MM-DD".

XSLT

XSLT (**Extensible Stylesheet Language Transformations**) is a declarative, XML-based language used for the transformation of XML documents. The original document is not changed; rather, a new document is created based on the content of an existing one.[2] The new

document may be serialized (output) by the processor in standard XML syntax or in another format, such as HTML or plain text.[3] XSLT is most often used to convert data between different XML schemas or to convert XML data into web pages or PDF documents.

Simple API for XML (SAX)

SAX is a lexical, event-driven interface in which a document is read serially and its contents are reported as callbacks to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

Document Object Model (DOM)

DOM (Document Object Model) is an interface-oriented Application Programming Interface that allows for navigation of the entire document as if it were a tree of "Node" objects representing the document's contents. A DOM document can be created by a parser, or can be generated manually by users (with limitations). Data types in DOM Nodes are abstract; implementations provide their own programming language-specific bindings. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed.

UNIT – IV

Over View :

This chapter provides an overview of an exciting technology that is at the forefront of Java programming: Java Beans. Beans are important, because they allow you to build complex systems from software components. These components may be provided by you or supplied by one or more different vendors. Java Beans defines an architecture that specifies how these building blocks can operate together.

Contents:

Introduction to Java Beans

Advantages of Java Beans

BDK

Introspection

- Bound properties
- Bean Info interface

Persistence

Customizers

Java beans API

Introduction to EJB

Introduction to Java Beans

As software developers, we are constantly being asked to build applications in less time and with less money. And, of course, these applications are expected to be better and faster than ever before. Object-oriented techniques and component software environments are in wide use now, in the hope that they can help us build applications more quickly. The JavaBeans architecture brings the component development model to Java.

A Java Bean is a reusable software component that can be manipulated visually in a builder tool. Beans will range greatly in their features and capabilities. Some will be very simple and others complex; some will have a visual aspect and others won't. Therefore, it isn't easy to put all Beans into a single category. Let's take a look at some of the most important features and issues surrounding Beans. This should set the stage for the rest of the book, where we will examine the JavaBeans technology in depth.

Advantages of Java Beans

- A Bean obtains all the benefits of Java's "write-once, run-anywhere" paradigm.
- The properties, events, and methods of a Bean that are exposed to an application builder tool can be controlled.
- A Bean may be designed to operate correctly in different locales, which makes it useful in global markets.
- Auxiliary software can be provided to help a person configure a Bean. This software is only needed when the design-time parameters for that component are being set. It does not need to be included in the run-time environment.
- The configuration settings of a Bean can be saved in persistent storage and restored at a later time.
- A Bean may register to receive events from other objects and can generate events that are sent to other objects.

The Bean Developer Kit (BDK)

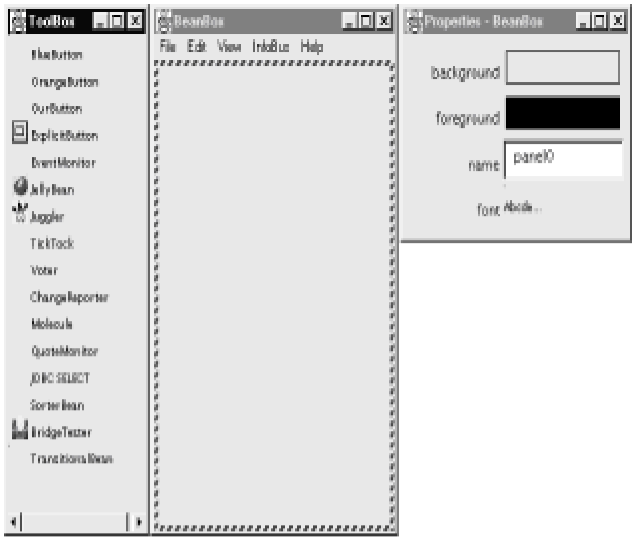
The Bean Developer Kit (BDK), available from the JavaSoft site, is a simple example of a tool that enables you to create, configure, and connect a set of Beans. There is also a set of sample Beans with their source code. This section provides step-by-step instructions for installing and using this tool.

Starting the BDK

To start the BDK, follow these steps:

1. Change to the directory **c:\\bdk\\beanbox**.

2. Execute the batch file called **run.bat**. This causes the BDK to display the three windows shown in Figure . ToolBox lists all of the different Beans that have been included with the BDK. BeanBox provides an area to lay out and connect the Beans selected from the ToolBox. Properties provides the ability to configure a selected Bean.



Create and Configure an Instance of the Molecule Bean

Follow these steps to create and configure an instance of the **Molecule** Bean:

1. Position the cursor on the ToolBox entry labeled **Molecule** and click the left mouse button. You should see the cursor change to a cross.

2. Move the cursor to the BeanBox display area and click the left mouse button in approximately the area where you wish the Bean to be displayed. You should see a rectangular region appear that contains a 3-D display of a molecule. This area is surrounded by a hatched border, indicating that it is currently selected.

3. You can reposition the **Molecule** Bean by positioning the cursor over one of the hatched borders and dragging the Bean.

4.You can change the molecule that is displayed by changing the selection in the Properties window. Notice that the Bean display changes immediately when you change the selected molecule.

Introspection

Introspection is the process of analyzing a Bean to determine its capabilities. This is an essential feature of the Java Beans API, because it allows an application builder tool to present information about a component to a software designer. Without introspection, the Java Beans technology could not operate.

There are two ways in which the developer of a Bean can indicate which of its properties, events, and methods should be exposed by an application builder tool. With the first method, simple naming conventions are used. These allow the introspection mechanisms to infer information about a Bean. In the second way, an additional class is provided that

explicitly supplies this information. The first approach is examined here. The second method is described later.

Design Patterns for Properties

Simple Properties

A simple property has a single value. It can be identified by the following design patterns, where N is the name of the property and T is its type. public T getN( );

public void setN(T arg);

Boolean Properties

A Boolean property has a value of **true** or **false**. It can be identified by the following design patterns, where N is the name of the property: public boolean isN( );

public boolean getN( );

public void setN(boolean value);

Indexed Properties

An indexed property consists of multiple values. It can be identified by the following design patterns, where N is the name of the property and T is its type: public T getN(int index);

public void setN(int index, T value);

public T[ ] getN( );

public void setN(T values[ ]);

Design Patterns for Events

Beans use the delegation event model that was discussed earlier in this book. Beans can generate events and send them to other objects.

These can be identified by the following design patterns, where T is the type of the event:

public void addTListener(TListener eventListener);

public void addTListener(TListener eventListener) throws
TooManyListeners; public void removeTListener(TListener eventListener);

Customizers

The Properties window of the BDK allows a developer to modify the properties of a Bean. However, this may not be the best user interface for a complex component with many interrelated properties. Therefore, a Bean developer can provide a *customizer* that helps another developer configure this software. A customizer can provide a step-by-step guide through the process that must be followed to use the component in a specific context. Online documentation can also be provided. A Bean developer has great flexibility

The Java Beans API

The Java Beans functionality is provided by a set of classes and interfaces in the **java.beans** package. This section provides a brief overview of its contents.

List of interfaces in **java.beans**.

AppletInitializer

Methods in this interface are used to initialize Beans that are a lso applets.

BeanInfo

This interface allows a designer to specify information about the properties, events, and

methods of a Bean.

Customizer

This interface allows a designer to provide a graphical user interface through which a Bean may be configured.

DesignMode

Methods in this interface determine if a Bean is executing in design mode.

PropertyChangeListener

A method in this interface is invoked when a bound property is changed.

PropertyEditor

Objects that implement this interface allow designers to change and display property values.

The Classes Defined in java.beans

BeanDescriptor

This class provides information about a Bean. It also allows you to associate a customizer with a Bean.

Beans

This class is used to obtain information about a Bean.

EventSetDescriptor

Instances of this class describe an event that can be generated by a Bean.

FeatureDescriptor

This is the superclass of the PropertyDescriptor, EventSetDescriptor, and MethodDescriptor classes.

IndexedPropertyDescriptor

Instances of this class describe an indexed property of a Bean.

IntrospectionException

An exception of this type is generated if a problem occurs when analyzing a Bean.

Introspector

This class analyzes a Bean and constructs a BeanInfo object that describes the component.

MethodDescriptor

I nstances of this class describe a method of a Bean.

ParameterDescriptor

I nstances of this class describe a method parameter.

PropertyChangeEvent

This event is generated when bound or constrained properties are changed. It is sent to objects that registered an interest in these events and implement either the PropertyChangeListener or VetoableChangeListener interfaces.

PropertyChangeSupport

Beans that support bound properties can use this class to notify PropertyChangeListener objects.

PropertyDescriptor

I nstances of this class describe a property of a Bean.

PropertyEditorManager

This class locates a PropertyEditor object for a given type.

PropertyEditorSupport

This class provides functionality that can be used when writing property editors.

Prope rtyVetoException

An exception of this type is generated if a change to a constrained property is vetoed.

SimpleBeanInfo

This class provides functionality that can be used when writing BeanInfo classes.

VetoableChangeSupport

Beans that support constrained properties can use this class to notify VetoableChangeListener objects.

EJB(Enterprise Java Beans)

- EJB's are simply Java classes coded as per EJB specification and deployed in a J2EE-compliant EJB container.
- It is a standard used for developing server-side business components.
- The intention behind this standard is to enable developing component-based, distributed enterprise applications.
- It is used to encapsulate the business logic or data logic of an application.
- EJB adopts a divide-and-conquer approach to server-side computing.
- Each component has a well-known but limited duty to be self-consistent.
- EJBs are not complete applications, but are deployable components that can be assembled into complete solutions.
- EJBs allow developers to concentrate on implementing business logic rather than low-level issues, since they are handled via container services.

EJB Architecture

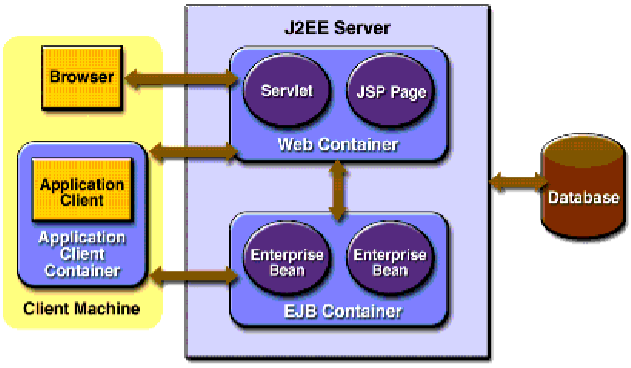
- EJBs are J2EE components that run in EJB container within an Application Server.
- Although transparent to application developers, EJB containers provide services (security, transactions) to its EJB's.

- These services enables quick development and deployment of enterprise.

- According to EJB specification, the EJB container, the enterprise beans and the client programs each have certain roles and responsibilities within the overall system, which is spelled in terms of contracts.

- These services enables quick development and deployment of enterprise.

- According to EJB specification, the EJB container, the enterprise beans and the client programs each have certain roles and responsibilities within the overall system, which is spelled in terms of contracts.



EJB Types

- **Stateless Session Beans** : executes business logic on behalf of any client. They usually perform relatively quick and simple tasks implementing control, process and workflow in an application. Ex: returning current stock price.
- **Stateful Session Beans**: They execute business logic on behalf of a specific client with whom they are maintaining a conversational state. Ex: online shopping cart, where user's purchases are tracked until checkout.
- **Entity Beans** : They manage database storage and retrieval. They are typically used in conjunction with session beans. Ex: Management of customer's stock portfolio.
- **Message Driven Beans:** New in EJB 2.0. They work with JMS implementation to provide asynchronous messaging based processing for any client that sends the message.

UNIT - V

Overview :

This unit describes how to generate dynamic content in webpages using servlets and procedure for installing web servers procedure for creating web applications.

Contents :

Introduction to servlets

Lifecycle of a servlet

JSDK server

Servlet API

Session tracking

Security issues

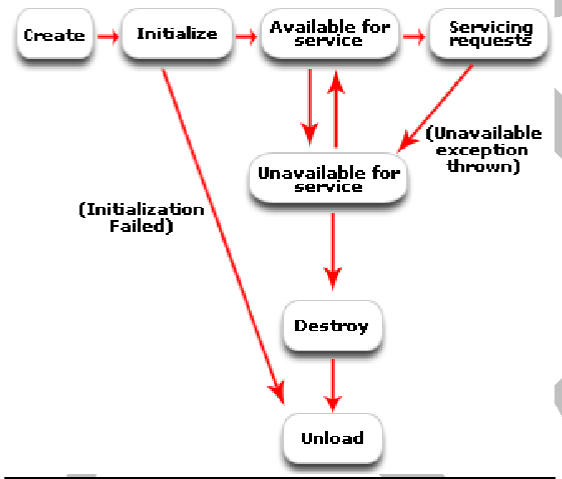
Introduction to servlets:

Servlets are java programs that run on web or application servers, acting as middle layer between request coming from the web browsers or other HTTP clients and database servers. Servlets process user request, produce the results and sends the results as a response to the user

Advantages of Java Servlets

1. Portability
2. Powerful
3. Efficiency
4. Safety
5. Extensibilty
6. Inexpensive

Life cycle of Servlet



- **Loading:** The servlet container loads the servlet during startup or when the first request is made. The loading of the servlet depends on the attribute <load-on-startup> of web.xml file. If the attribute <load-on-startup> has a positive value then the servlet is load with loading of the container otherwise it load when the first request comes for service. After loading of the servlet, the container creates the instances of the servlet.

- **Initialization:** After creating the instances, the servlet container calls the init() method and passes the servlet initialization parameters to the init() method. The init() must be called by the servlet container before the servlet can service any request. The initialization parameters persist untill the servlet is destroyed. The init() method is called only once throughout the life cycle of the servlet.

  The servlet will be available for service if it is loaded successfully otherwise the servlet container unloads the servlet.

- **Servicing the Request:** After successfully completing the initialization process, the servlet will be available for service. Servlet creates seperate threads for each request. The sevlet container calls the service() method for servicing any request. The service() method determines the kind of request and calls the appropriate method (doGet() or doPost()) for handling the request and sends response to the client using the methods of the response object.

- **Destroying the Servlet:** If the servlet is no longer needed for servicing any request, the servlet container calls the destroy() method . Like the init() method this method is also called only once throughout the life cycle of the servlet. Calling the destroy() method indicates to the servlet container not to sent the any request for service and the servlet

releases all the resources associated with it. Java Virtual Machine claims for the memory associated with the resources for garbage collection.

Servlet API Provides the following two packages

- **Javax.servlet**

  The javax.servlet package contains a number of classes and interfaces that describe and define the contracts between a servlet class and the runtime environment provided for an instance of such a class by a conforming servlet container.

- **Javax.servlet.http**

The javax.servlet.http package contains a number of classes and interfaces that describe and define the contracts between a servlet class running under the HTTP protocol and the runtime environment provided for an instance of such a class by a conforming servlet container.

1) **javax.servlet.Servlet interface**

- A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.
- To implement this interface, you can write a generic servlet that extends

  `javax.servlet.GenericServlet` or an HTTP servlet that extends `javax.servlet.http.HttpServlet`.

- This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server.

- In addition to the life-cycle methods(init(),service(),destroy()), this interface provides the `getServletConfig` method, which the servlet can use to get any startup information, and the `getServletInfo` method, which allows the servlet to return basic information about itself, such as author, version, and copyright.

Methods:

- **init**

public void **init**(ServletConfig config)

throws ServletException

Called by the servlet container to indicate to a servlet that the servlet is being placed into service.

The servlet container calls the `init` method exactly once after instantiating the servlet. The `init` method must complete successfully before the servlet can receive any requests.

The servlet container cannot place the servlet into service if the `init` method

1. Throws a ServletException
2. Does not return within a time period defined by the Web server

Parameters:

config - a ServletConfig object containing the servlet's configuration and initialization

parameters

Throws:

ServletException - if an exception has occurred that interferes with the servlet's normal operation

getServletConfig

public ServletConfig **getServletConfig**()

Returns a ServletConfig object, which contains initialization and startup parameters for this servlet. The ServletConfig object returned is the one passed to the init method.

- **service**

public void **service**(ServletRequest req,ServletResponse res) throws ServletException, java.io.IOException

Called by the servlet container to allow the servlet to respond to a request.

This method is only called after the servlet's `init()` method has completed successfully.

Parameters:

req - the ServletRequest object that contains the client's request res
- the ServletResponse object that contains the servlet's response

Throws:

ServletException - if an exception occurs that interferes with the servlet's normal

operation

java.io.IOException - if an input or output exception occurs

- **getServletInfo**

public java.lang.String **getServletInfo**()

Returns information about the servlet, such as author, version, and copyright.

- **destroy**

public void **destroy**()

Called by the servlet container to indicate to a servlet that the servlet is being taken out of service. This method is only called once all threads within the servlet's service method have exited or after a timeout period has passed. After the servlet container calls this method, it will not call the service method again on this servlet.

This method gives the servlet an opportunity to clean up any resources that are being held (for example, memory, file handles, threads) and make sure that any persistent state is synchronized with the servlet's current state in memory.

2) javax.servlet.ServletConfig Interface

A servlet configuration object used by a servlet container to pass information to a servlet during initialization.

Methods:

- **getServletName**

public java.lang.String **getServletName**()

Returns the name of this servlet instance. The name may be provided via server administration, assigned in the web application deployment descriptor, or for an unregistered (and thus unnamed) servlet instance it will be the servlet's class name.

- **getServletContext**

public ServletContext **getServletContext**()

Returns a reference to the ServletContext in which the caller is executing.

Returns:

a ServletContext object, used by the caller to interact with its servlet container

- **getInitParameter**

public java.lang.String **getInitParameter**(java.lang.String name)

Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.

Parameters:

name - a String specifying the name of the initialization parameter

Returns:

a String containing the value of the initialization parameter

- **getInitParameterNames**

public java.util.Enumeration **getInitParameterNames**()

Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.
Returns: an Enumeration of String objects containing the names of the servlet's initialization parameters

3) javax.servlet.ServletContext Interface

Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file

There is one context per "web application" per Java Virtual Machine
The ServletContext object is contained within the ServletConfig object.

Mehods:

java.lang.String **getInitParameter**(java.lang.String name)

Returns a String containing the value of the named context-wide initialization parameter,

or null if the parameter does not exist.

java.util.Enumeration **getInitParameterNames**()

Returns the names of the context's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the context has no initialization parameters.
java.lang.String **getMimeType**(java.lang.String file)

Returns the MIME type of the specified file, or null if the MIME type is not known. RequestDispatcher **getNamedDispatcher**(java.lang.String name)

Returns a RequestDispatcher object that acts as a wrapper for the named servlet. RequestDispatcher **getRequestDispatcher**(java.lang.String path)

Returns a RequestDispatcher object that acts as a wrapper for the resource located at the given path

java.lang.String **getServerInfo**()

Returns the name and version of the servlet container on which the servlet is running.

java.lang.String **getServletContextName**()

Returns the name of this web application corresponding to this ServletContext as specified in the deployment descriptor for this web application by the display-name element.

Void **setAttribute**(java.lang.String name, java.lang.Object object)
Binds an object to a given attribute name in this servlet context.

java.lang.Object **getAttribute**(java.lang.String name)

Returns the servlet container attribute with the given name, or null if there is no attribute by that name.

4) javax.servlet. RequestDispatcher Interface

Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server

This interface is intended to wrap servlets, but a servlet container can create RequestDispatcher objects to wrap any type of resource

Methods:

- **forward**

public void **forward**(ServletRequest request,

ServletResponse response)

throws ServletException,

java.io.IOException

Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server

- **include**

public void **include**(ServletRequest request,

ServletResponse response)

throws ServletException,

java.io.IOException

Includes the content of a resource (servlet, JSP page, HTML file) in the response. In essence, this method enables programmatic server-side includes.

5) **javax.servlet.ServletRequest Interface**

Defines an object to provide client request information to a servlet. The servlet container creates a ServletRequest object and passes it as an argument to the servlet's service method

A ServletRequest object provides data including parameter name and values, attributes, and an input stream.

1) **javax.servlet.ServletResponse Interface**

Defines an object to assist a servlet in sending a response to the client. The servlet container creates a `ServletResponse` object and passes it as an argument to the servlet's `service` method.

To send binary data in a MIME body response, use the ServletOutputStream returned by getOutputStream().

To send character data, use the PrintWriter object returned by getWriter().

1) **javax.servlet.GenericServlet class**

Defines a generic, protocol-independent servlet.

`GenericServlet` implements the `Servlet` and `ServletConfig` interfaces.

`GenericServlet` may be directly extended by a servlet, although it's more common to extend a protocol-specific subclass such as `HttpServlet`.

Interfaces and classes of javax.servlet.http package

1. javax.servlet.http. HttpServletRequest Interface

Extends the `ServletRequest` interface to provide request information for HTTP servlets.

The servlet container creates an `HttpServletRequest` object and passes it as an argument to the servlet's service methods (`doGet`, `doPost`, etc).

Methods

- **getCookies**

public Cookie[] **getCookies**()

Returns an array containing all of the Cookie objects the client sent with this request.

This method returns null if no cookies were sent.

- **getSession**

public HttpSession **getSession**()

Returns the current session associated with this request, or if the request does not have a session, creates one.

public HttpSession **getSession**(boolean create)

Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

If `create` is `false` and the request has no valid `HttpSession`, this method returns `null`.

- **getRequestURI**

public java.lang.String **getRequestURI**()

Returns the part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.

For example POST /some/path.html HTTP/1.1 then it returns /some/path.html

- **getRequestURL**

public java.lang.StringBuffer **getRequestURL**()

Reconstructs the URL the client used to make the request. The returned URL contains a protocol, server name, port number, and server path, but it does not include query string parameters.

- **getServletPath**

public java.lang.String **getServletPath**()

Returns the part of this request's URL that calls the servlet. This path starts with a "/" character and includes either the servlet name or a path to the servlet, but does not include any extra path information or a query string.

- **getContextPath**

public java.lang.String **getContextPath**()

Returns the portion of the request URI that indicates the context of the request. The context path always comes first in a request URI. The path starts with a "/" character but does not end with a "/" character.

- **getHeader**

public java.lang.String **getHeader**(java.lang.String name)

Returns the value of the specified request header as a String.

- **getHeaders**

public java.util.Enumeration **getHeaders**(java.lang.String name)

Returns all the values of the specified request header as an Enumeration of String objects.

- **getHeaderNames**

public java.util.Enumeration **getHeaderNames**()

Returns an enumeration of all the header names this request contains. If the request has no headers, this method returns an empty enumeration.

2) javax.servlet.http. HttpServletResponse Interface

Extends the `ServletResponse` interface to provide HTTP-specific functionality in sending a response. For example, it has methods to access HTTP headers and cookies.

The servlet container creates an HttpServletResponse object and passes it as an argument to the servlet's service methods

- **addCookie**

public void **addCookie**(Cookie cookie)

Adds the specified cookie to the response. This method can be called multiple times to set more than one cookie

- **sendError**

public void **sendError**(int sc)

throws java.io.IOException

Sends an error response to the client using the specified status code and clearing the buffer.

If the response has already been committed, this method throws an IllegalStateException

- **sendRedirect**

public void **sendRedirect**(java.lang.String location)

throws java.io.IOException

Sends a temporary redirect response to the client using the specified redirect location URL. This method can accept relative URLs; location - the redirect location URL

- **setHeader**

public void **setHeader**(java.lang.String name,

java.lang.String value)

Sets a response header with the given name and value. If the header had already been set, the new value overwrites the previous one.

- **addHeader**

public void **addHeader**(java.lang.String name,

java.lang.String value)

Adds a response header with the given name and value. This method allows response headers to have multiple values.

- **addIntHeader**

public void **addIntHeader**(java.lang.String name, int value)

Adds a response header with the given name and integer value. This method allows response headers to have multiple values.

- **setStatus**

public void **setStatus**(int sc)

Sets the status code for this response. This method is used to set the return status code when there is no error

3) javax.servlet.http. HttpSession Interface

HttpSession provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

A session usually corresponds to one user, who may visit a site many times. The server can maintain a session in many ways such as using cookies or rewriting URLs.

This interface allows servlets to

- View and manipulate information about a session, such as the session identifier, creation time, and last accessed time
- Bind objects to sessions, allowing user information to persist across multiple user connections

Methods:

- **getAttribute**

public java.lang.Object **getAttribute**(java.lang.String name)

Returns the object bound with the specified name in this session, or null if no object is bound under the name.

- **setAttribute**

public void **setAttribute**(java.lang.String name,

java.lang.Object value)

Binds an object to this session, using the name specified. If an object of the same name is already bound to the session, the object is replaced.

- **getAttributeNames**

public java.util.Enumeration **getAttributeNames**()

Returns an Enumeration of String objects containing the names of all the objects bound to this session.

- **removeAttribute**

public void **removeAttribute**(java.lang.String name)

Removes the object bound with the specified name from this session.

- **invalidate**

public void **invalidate**()

Invalidates this session then unbinds any objects bound to it.

- **isNew**

public boolean **isNew**()

returns true if the server has created a session, but the client has not yet joined

- **getMaxInactiveInterval**

public int **getMaxInactiveInterval**()

Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.

- **setMaxInactiveInterval**

public void **setMaxInactiveInterval**(int interval)

Specifies the time, in seconds, between client requests before the servlet container will invalidate this session. A negative time indicates the session should never timeout.

- **getLastAccessedTime**

public long **getLastAccessedTime**()

Returns the last time the client sent a request associated with this session,

- **getId**

public java.lang.String **getId**()

Returns a string containing the unique identifier assigned to this session.

- **getCreationTime**

public long **getCreationTime**()

Returns the time when this session was created

Session Tracking:

Http is a *stateless* protocol, means that it can't persist the information. It always treats each request as a new request. In Http client makes a connection to the server, sends the request., gets the response, and closes the connection.

In session management client first make a request for any servlet or any page, the container receives the request and generate a unique session ID and gives it back to the client along with the response. This ID gets stores on

the client machine. Thereafter when the client request again sends a request to the server then it also sends the session Id with the request. There the container sees the Id and sends back the request.

Session Tracking can be done in Four ways:

1. **Hidden Form Fields:**
2. **Cookies**
3. **HttpSession**
4. **URL Rewriting**

UNIT - VI

Overview :

In this unit we focuses on JSP Technologies. JavaServer Pages (JSP) is a Java technology that helps software developers serve dynamically generated web pages based on HTML, XML, or other document types. JSP pages are loaded in the server and are operated from a structured special installed Java server packet called a Java EE Web Application, often packaged as a `.war` or `.ear` file archive. JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, with the resulting page being compiled and executed on the server to deliver an HTML or XML document. The compiled pages and any dependent Java libraries use Java bytecode rather than a native software format, and must therefore be executed within a Java virtual machine (JVM) that integrates with the host operating system to provide an abstract platform-neutral environment.

Contents

Problems with servlets

The Anatomy of JSP Page

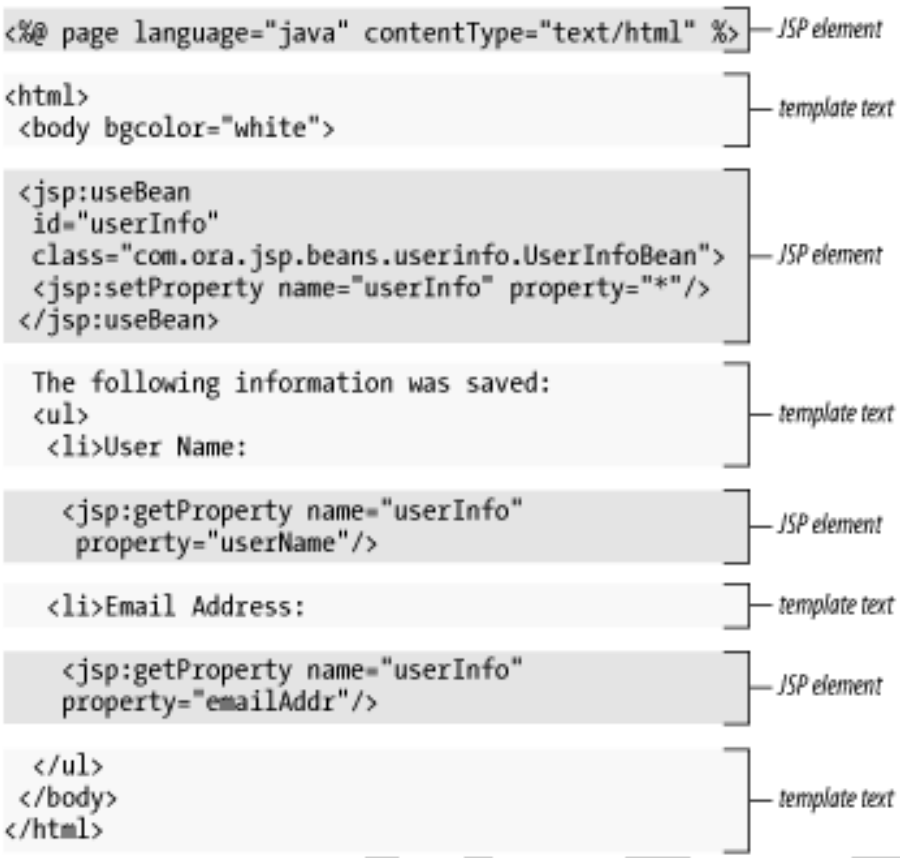
JSP Processing

MVC

JSDK

The Anatomy of a JSP Page

A JSP page is simply a regular web page with JSP elements for generating the parts that differ for each request,



Everything in the page that isn't a JSP element is called *template text*. Template text can be any text: HTML, WML, XML, or even plain text. Since HTML is by far the most common web-

page language in use today, most of the descriptions and examples in this book use HTML, but keep in mind that JSP has no dependency on HTML; it can be used with any markup language. Template text is always passed straight through to the browser.

When a JSP page request is processed, the template text and dynamic content generated by the JSP elements are merged, and the result is sent as the response to the browser.

JSP Processing

Just as a web server needs a servlet container to provide an interface to servlets, the server needs a *JSP container* to process JSP pages.

The JSP container is responsible for intercepting requests for JSP pages. To process all JSP elements in the page, the container first turns the JSP page into a servlet (known as the *JSP page implementation class*).

The conversion is pretty straightforward; all template text is converted to `println( )` statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior. The container then compiles the servlet class.

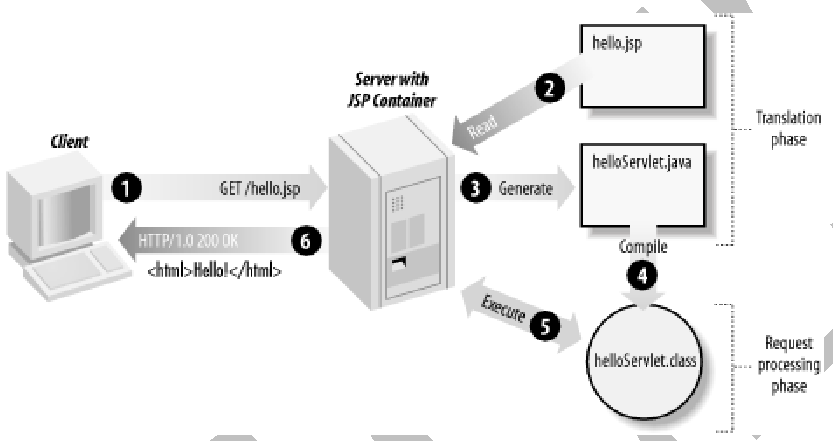
Converting the JSP page to a servlet and compiling the servlet form the *translation phase*.

The JSP container initiates the translation phase for a page automatically when it receives the first request for the page. Since the translation phase takes a bit of time, the first user to request a JSP page notices a slight delay.

The translation phase can also be initiated explicitly; this is referred to as *precompilation* of a JSP page. Precompiling a JSP page is a way to avoid hitting the first user with this delay.

The JSP container is also responsible for invoking the JSP page implementation class (the generated servlet) to process each request and generate the response. This is called the *request processing phase*.

JSP page translation and processing phases



As long as the JSP page remains unchanged, any subsequent request goes straight to the request processing phase (i.e., the container simply executes the class file). When the JSP page is modified, it goes through the translation phase again before entering the request processing phase.

The JSP container is often implemented as a servlet configured to handle all requests for JSP pages. In fact, these two containers--a servlet container and a JSP container--are often combined in one package under the name *web container*. a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet: it's loaded once and called repeatedly, until the server is shut down. By virtue of being an automatically generated servlet, a JSP page inherits all the advantages of a servlet: platform and vendor independence, integration, efficiency, scalability, robustness, and security.

JSP Application Design with MVC

. The key point of using MVC is to separate logic into three distinct units: the Model, the View, and the Controller.

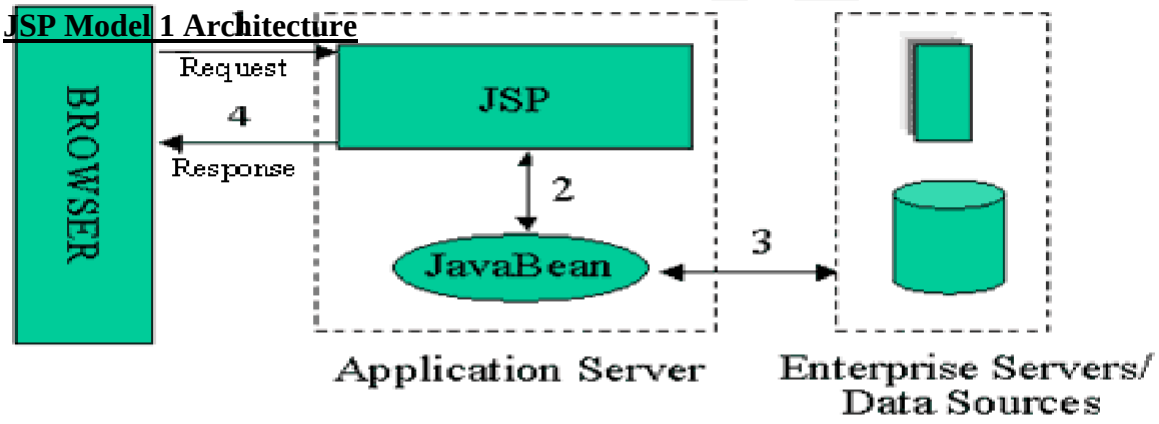
In a server application, we commonly classify the parts of the application as business logic, presentation, and request processing.

Business logic is the term used for the manipulation of an application's data, such as customer, product, and order information.

Presentation refers to how the application data is displayed to the user, for example, position, font, and size.

And finally, *request processing* is what ties the business logic and presentation parts together.

JSP Model 1 Architecture



In MVC terms, the Model corresponds to business logic and data, the View to the presentation, and the Controller to the request processing.

Why use this design with JSP? The answer lies primarily in the first two elements. Remember that an application data structure and logic (the Model) is typically the most stable part of an application, while the presentation of that data (the View) changes fairly often. Just look at all the face-lifts many web sites go through to keep up with the latest fashion in web design. Yet, the data they present remains the same.

Another common example of why presentation should be separated from the business logic is that you may want to present the data in different languages or present different subsets of the data to internal and external users.

Access to the data through new types of devices, such as cell phones and personal digital assistants (PDAs), is the latest trend. Each client type requires its own presentation format. It should come as no surprise, then, that separating business logic from the presentation makes it easier to evolve an application as the requirements change; new presentation interfaces can be developed without touching the business logic.

JSDK

1 Set classpath to <home dir>/jsdk2.0/lib/jsdk.jar;

2. Set path to <home dir>/jsdk2.0/bin;

3. Compile the servlet program (.java file).

4. Copy the .class file to <home dir>/jsdk2.0/examples

5. Execute servletrunner

6. Open web browser

7. Enter the url as follows in the address bar: http://host:8080/servlet/<servlet name>

UNIT - VII

Overview :

In this unit we focus on how to develop applications using JSP. JSP syntax is a fluid mix of two basic content forms: *scriptlet elements* and *markup*. Markup is typically standard HTML or XML, while scriptlet elements are delimited blocks of Java code which may be intermixed with the markup. When the page is requested the Java code is executed and its output is added, in situ, with the surrounding markup to create the final page. JSP pages must be compiled to Java bytecode classes before they can be executed, but such compilation is needed only when a change to the source JSP file has occurred.

Contents

JSP elements

JSP objects

Sharing data between JSP pages

Error Handling and Debugging

JSP Elements

There are three types of JSP elements you can use: *scripting*, *action* and *directive*.

Scripting elements

Scripting elements allow you to add small pieces of code (typically Java code) in a JSP page, such as an `if` statement to generate different HTML depending on a certain condition.

Like actions, they are also executed when the page is requested. You should use scripting elements with extreme care: if you embed too much code in your JSP pages, you will end up with the same kind of maintenance problems as with servlets embedding HTML.

Scripting elements

Element	Description
`<% ... %>`	Scriptlet, used to embed scripting code.
`<%= ... %>`	Expression, used to embed scripting code expressions when the result shall be added to the response. Also used as request-time action attribute values.
`<%! ... %>`	Declaration, used to declare instance variables and methods in the JSP page implementation class.

Standard action elements

Action elements typically perform some action based on information that is required at the exact time the JSP page is requested by a browser. An action can, for instance, access parameters sent with the request to do a database lookup. It can also dynamically generate HTML, such as a table filled with information retrieved from an external system.

The JSP specification defines a few standard action elements

Action element	Description

`<jsp:useBean>`	Makes a JavaBeans component available in a page
`<jsp:getProperty>`	Gets a property value from a JavaBeans component and adds it to the response
`<jsp:setProperty>`	Sets a JavaBeans component property value
`<jsp:include>`	Includes the response from a servlet or JSP page during the request processing phase
`<jsp:forward>`	Forwards the processing of a request to servlet or JSP page
`<jsp:param>`	Adds a parameter value to a request handed off to another servlet or JSP page using `<jsp:include>` or `<jsp:forward>`
`<jsp:plugin>`	Generates HTML that contains the appropriate browser-dependent elements (OBJECT or EMBED) needed to execute an applet with the Java Plug-in software

Custom action elements and the JSP Standard Tag Library

In addition to the standard actions, the JSP specification includes a Java API a programmer can use to develop *custom actions* to extend the JSP language. The JSP Standard Tag Library (JSTL) is such an extension, with the special status of being defined by a formal specification from Sun and typically bundled with the JSP container. JSTL contains action elements for processes needed in most JSP applications, such as conditional processing, database access, internationalization, and more.

If JSTL isn't enough, programmers on your team (or a third party) can use the extension API to develop additional custom actions, may be to access application-specific resources or simplify application-specific processing.

JavaBeans components

JSP elements, such as action and scripting elements, are often used to work with JavaBeans. Put succinctly, a JavaBeans component is a Java class that complies with certain coding conventions. JavaBeans components are typically used as containers for information that describes application entities, such as a customer or an order.

Directive elements

The directive elements specify information about the page itself that remains the same between requests--for example, if session tracking is required or not, buffering requirements, and the name of a page that should be used to report errors, if any.

Directive elements

Element	Description
`<%@ page ... %>`	Defines page-dependent attributes, such as session tracking, error page, and buffering requirements
`<%@ include ... %>`	

`<%@ taglib ... %>`	Includes a file during the translation phase
	Declares a tag library, containing custom actions, that is used in the page

JSP objects

JSP object	Servlet API Object	Description
application	javax.servlet.ServletContext	Context (Execution environment) of the Servlet.

config	javax.servlet.ServletConfig	The ServletConfig for the JSP.
exception	java.lang.Throwable	The exception that resulted when an error occurred.
out	javax.servlet.jsp.JspWriter	An object that writes into a JSP's output stream.

pageContext	javax.servlet.jsp.PageContext	Page context for the JSP.
request	javax.servlet.HttpServletRequest	The client request.
response	javax.servlet.HttpServletResponse	The response to the client.
session	javax.servlet.http.HttpSession	Session object created for requesting client.
page	javax.servlet.Servlet	Refers to current servlet object.

Error Handling and Debugging

When you develop any application that's more than a trivial example, errors are inevitable. A JSP-based application is no exception. There are many types of errors you will deal with.

Simple syntax errors in the JSP pages are almost a given during the development phase. And even after you have fixed all the syntax errors, you may still have to figure out why the application doesn't work as you intended because of design mistakes. The application must also be designed to deal with problems that can occur when it's deployed for production use. Users can enter invalid values and try to use the application in ways you never imagined. External systems, such as databases, can fail or become unavailable due to network problems.

Since a web application is the face of the company, making sure it behaves well, even when the users misbehave and the world around it falls apart, is extremely important for a positive customer perception. Proper design and testing is the only way to accomplish this goal.

Sharing Data Between JSP Pages, Requests, and Users

- **Passing Control and Data Between Pages**

one of the most fundamental features of JSP technology is that it allows for separation of request processing, business logic and presentation, using what's known as the Model-View-

Controller (MVC) model. As you may recall, the roles of Model, View, and Controller can be assigned to different types of server-side components. In this part of the book, JSP pages are used for both the Controller and View roles, and the Model role is played by either a bean or a JSP page.

- **Sharing Session and Application Data**

The request scope makes data available to multiple pages processing the same request. But in many cases, data must be shared over multiple requests.

Imagine a travel agency application. It's important to remember the dates and destination entered to book the flight so that the customer doesn't have to reenter the information when it's time to make hotel and rental car reservations. This type of information, available only to requests from the same user, can be shared through the session scope.

Memory Usage Considerations

You should be aware that all objects you save in the application and session scopes take up memory in the server process. It's easy to calculate how much memory is used for the application scope because you have full control over the number of objects you place there. But the total number of objects in the session scope depends on the number of concurrent sessions, so in addition to the size of each object, you also need to know how many concurrent users you have and how long a session lasts. Let's look at an example.

The CartBean used in this chapter is small. It stores only references to ProductBean instances, not copies of the beans. An object reference in Java is 8 bytes, so with three products in the cart we need 24 bytes. The java.util.Vector object used to hold the references adds some overhead, say 32 bytes. All in all, we need 56 bytes per shopping cart bean with three products.

UNIT - VIII

Overview :

In this unit we focus on database access in simple java programs, servlets ,and JSPs. We ca access the database by using JDBC. JDBC stands for "Java DataBase Connectivity". It is an API which consists of a set of Java classes, interfaces and exceptions and a specification to which both JDBC driver vendors and JDBC developers adhere when developing applications.

Contents

JDBC drivers

Javax.sql API

Database from JSP Page

Struts frame work

JDBC is a very popular data access standard. RDBMS (Relational Database Management Systems) or third-party vendors develop drivers which adhere to the JDBC specification. Other developers use these drivers to develop applications which access those databases e.g. you'll use ConnectorJ JDBC driver to access MySQL database. Since the drivers adhered to JDBC specification, the JDBC application developers can replace one driver for their application with another better one without having to rewrite their application. If they had used some proprietary API provided by some RDBMS vendor, they will not have been able to change the driver and/or database without having to rewrite the complete application.

JDBC Driver Types

JDBC drivers are divided into four types or levels. The **different types of jdbc drivers** are:

Type 1: JDBC-ODBC Bridge driver (Bridge)

Type 2: Native-API/partly Java driver (Native)

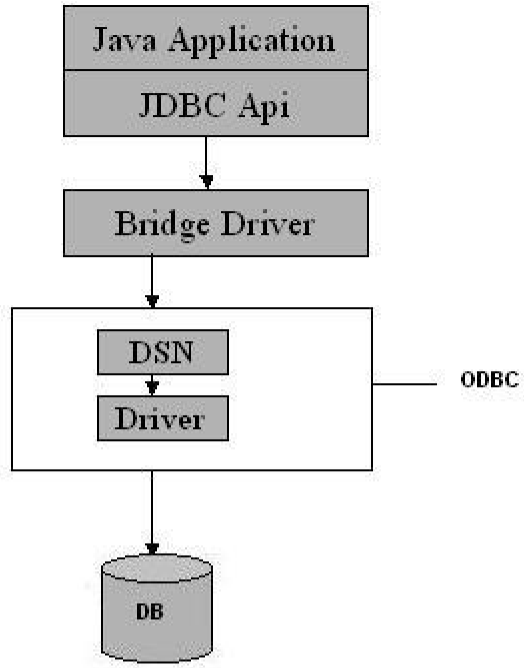
Type 3: AllJava/Net-protocol driver (Middleware)

Type 4: All Java/Native-protocol driver (Pure)

Type 1 JDBC Driver

JDBC-ODBC Bridge driver

The Type 1 driver translates all JDBC calls into ODBC calls and sends them to the ODBC driver. ODBC is a generic API. The JDBC-ODBC Bridge driver is recommended only for experimental use or when no other alternative is available.



Type 1: JDBC-ODBC Bridge

Advantage

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

Disadvantages

1. Since the Bridge driver is not written fully in Java, Type 1 drivers are not portable.

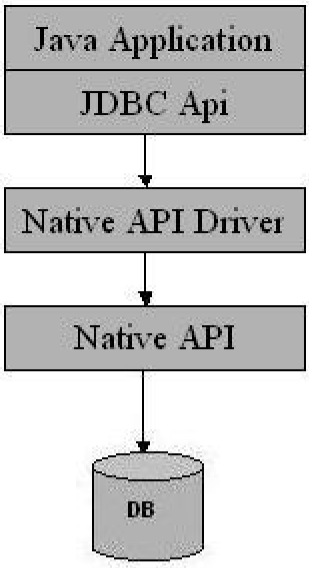
2. A performance issue is seen as a JDBC call goes through the bridge to the ODBC driver, then to the database, and this applies even in the reverse process. They are the slowest of all driver types.

3. The client system requires the ODBC Installation to use the driver.
4. Not good for the Web.

Type 2 JDBC Driver

Native-API/partly Java driver

The distinctive characteristic of type 2 jdbc drivers are that Type 2 drivers convert JDBC calls into database-specific calls i.e. this driver is specific to a particular database. Some distinctive characteristic of type 2 jdbc drivers are shown below. Example: Oracle will have oracle native api.



Type 2: Native api/ Partly Java Driver

Advantage

The distinctive characteristic of type 2 jdbc drivers are that they are typically offer better performance than the JDBC-ODBC Bridge as the layers of communication (tiers) are less than that of Type

1 and also it uses Native api which is Database specific.

Disadvantage

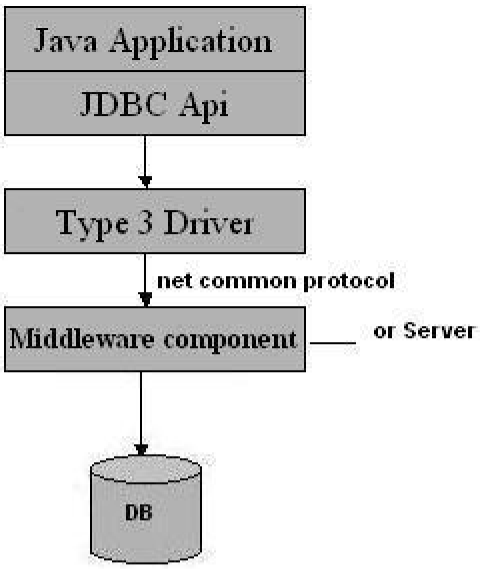
1. Native API must be installed in the Client System and hence type 2 drivers cannot be used for the Internet.

2. Like Type 1 drivers, it's not written in Java Language which forms a portability issue.
3. If we change the Database we have to change the native api as it is specific to a database
4. Mostly obsolete now
5. Usually not thread safe.

Type 3 JDBC Driver

All Java/Net-protocol driver

Type 3 database requests are passed through the network to the middle-tier server. The middle-tier then translates the request to the database. If the middle-tier server can in turn use Type1, Type 2 or Type 4 drivers.



Type 3: All Java/ Net-Protocol Driver

Advantage

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

2. This driver is fully written in Java and hence Portable. It is suitable for the web.
3. There are many opportunities to optimize portability, performance, and scalability.
4. The net protocol can be designed to make the client JDBC driver very small and fast to load.

5. The type 3 driver typically provides support for features such as caching (connections, query results, and so on), load balancing, and advanced

system administration such as logging and auditing.

6. This driver is very flexible allows access to multiple databases using one driver.
7. They are the most efficient amongst all driver types.

Disadvantage

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

Type 4 JDBC Driver

Native-protocol/all-Java driver

The Type 4 uses java networking libraries to communicate directly with the database server.

Type 4: Native-protocol/all-Java driver

Advantage

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

2. Number of translation layers is very less i.e. type 4 JDBC drivers don't have to translate database requests to ODBC or a native connectivity interface or to pass the request on to another server, performance is typically quite good.

3. You don’t need to install special software on the client or server. Further, these drivers can be downloaded dynamically.

Disadvantage

With type 4 drivers, the user needs a different driver for each database.

List of Drivers

- Bridge driver
    - sun.jdbc.odbc.JdbcOdbcDriver
        - jdbc:odbc:<dsn>
- Cloudscape
    - COM.cloudscape.core.JDBCDriver
        - jdbc:cloudscape:[database name and location]
- PostGRESQL
    - org.postgresql.Driver
        - jdbc:postgresql://[host]:[port]/[database name]
- MySQL
    - com.mysql.jdbc.Driver
        - jdbc:mysql://[host]:3306/[databasename]
- Oracle
    - oracle.jdbc.driver.OracleDriver

    jdbc:oracle:thin:@[host]:1521:[sid]

Steps to using Bridge driver

1. Create a data source name using ODBC
2. Load the database driver
3. Establish a Connection to the database
4. Create a Statement object

5. Execute SQL Query statement(s)
6. Retrieve the ResultSet Object
7. Retrieve record/field data from ResultSet
8. object for processing
9. Close ResultSet Object
10. Close Statement Object
11. Close Connection Object

javax.sql.RowSet

The interface that adds support to the JDBC API for the JavaBeans™ component model. A rowset, which can be used as a JavaBeans component in a visual Bean development environment, can be created and configured at design time and executed at run time.

The RowSet interface provides a set of JavaBeans properties that allow a RowSet instance to be configured to connect to a JDBC data source and read some data from the data source. A group of setter methods (setInt, setBytes, setString, and so on) provide a way to pass input parameters to a rowset's command property. This command is the SQL query the rowset uses when it gets its data from a relational database, which is generally the case.

The RowSet interface supports JavaBeans events, allowing other components in an application to be notified when an event occurs on a rowset, such as a change in its value.

javax.sql.DataSource

A factory for connections to the physical data source that this DataSource object represents. An alternative to the DriverManager facility, a DataSource object is the preferred means of getting a connection. An object that implements the DataSource interface will typically be registered with a naming service based on the Java™ Naming and Directory (JNDI) API.

The DataSource interface is implemented by a driver vendor. There are three types of implementations:

1. Basic implementation -- produces a standard Connection object
2. Connection pooling implementation -- produces a Connection object that will automatically participate in connection pooling. This implementation works with a middle-tier connection pooling manager.
3. Distributed transaction implementation -- produces a Connection object that may be used for distributed transactions and almost always participates in connection pooling. This implementation works with a middle-tier transaction manager and almost always with a connection pooling manager.

A DataSource object has properties that can be modified when necessary. For example, if the data source is moved to a different server, the property for the server can be changed. The benefit is that because the data source's properties can be changed, any code accessing that data source does not need to be changed.

A driver that is accessed via a DataSource object does not register itself with the DriverManager. Rather, a DataSource object is retrieved though a lookup operation and then used to create a Connection object. With a basic implementation, the connection obtained through a DataSource object is identical to a connection obtained through the DriverManager facility.

Specific database actions

```
<sql: transaction>

  <sql: update>

    UPDATE Account SET Balance = Balance -
      1000 WHERE AccountNumber = 1234

  </sql: update>

  <sql: update>

    UPDATE Account SET Balance = Balance +
      1000 WHERE AccountNumber = 5678

  </sql: update>

</sql: transaction>
```

All SQL actions that make up a transaction are placed in the body of a <sql:transaction> action element. This action tells the nested elements which database to use, so if you need to specify the database with the dataSource attribute, you must specify it for the <sql:transaction> action. The isolation attribute can specify special transaction features. When the Data Source is made available to the application through JNDI or by another application component, it's typically already configured with an appropriate isolation level. This attribute is therefore rarely used. The details of the different isolation levels are beyond the scope of this book. If you believe you need to specify this value, you can read up on the differences in the JDBC API documents or in the documentation for your database. You should also be aware that some databases and JDBC drivers don't support all transaction isolation levels.

Struts framework

Apache Struts is a free open-source framework for creating Java web applications. Web applications differ from conventional websites in that web applications can create a dynamic response. Many websites deliver only static pages. A web application can interact with databases and business logic engines to customize a response. Web applications based on JavaServer Pages sometimes commingle database code, page design code, and control flow

code. In practice, we find that unless these concerns are separated, larger applications become difficult to maintain.

One way to separate concerns in a software application is to use a Model-View-Controller (MVC) architecture. The *Model* represents the business or database code, the *View* represents the page design code, and the *Controller* represents the navigational code. The Struts framework is designed to help developers create web applications that utilize a MVC architecture.

The framework provides three key components:

- A "request" handler provided by the application developer that is mapped to a standard URI.

- A "response" handler that transfers control to another resource which completes the response.
- A tag library that helps developers create interactive form-based applications with server pages.

The framework's architecture and tags are buzzword compliant. Struts works well with conventional REST applications and with nouveau technologies like SOAP and AJAX.

Configuring for Struts

1. Download the Struts binary release from http://jakarta.apache.org.
2. Extract the zip file.
3. Copy the .jar files to lib directory of the Web application.
4. Copy the .tld files to WEB-INF directory.
5. Store web.xml and struts-config.xml files in WEB-INF directory. (struts-config.xml is the DD for all Struts applications. It links all MVC components).

ONLINE QUESTIONS

UNIT-I

Questions	opt1	opt2	opt3	opt4	opt 5	opt 6	answer
Java was first developed in ?	1990	1991	1993	1996			1991
The old name of Java was ?	J language	oak	oct	None of above			oak
Which of the following feature is not supported by java ?	Multithreading	Reflection	Operator Overloading	Garbage Collection			Operator Overloading
Which of the following is not keyword in java ?	null	import	volatile	package			null
What is the full form of JDK ?	Java Data Kit	Java Defination Kit	Java Development Kit	Java Design Kit			Java Development Kit
Which command is used to compile a java program ?	javac	java	javad	.javadoc			javac
What is the full form of JVM	Java Virtual Machine	Java Variable Machine	Java Virtual Mechanism	Java Variable Mechanism			Java Virtual Machine

What is the full form of ADT ?	Abstract Data Type	Abstract Developmen t tool	Abstract Design Tool	Advance Development Tool			Abstract Data Type
The expected signature of the main method is public static void main(). What happens if we make a mistake and forget to put the static keyword ?	The JVM issues an error saying that main method should be declared static	The compiler issues a warning saying that main method should be declared static and adds it automation	The JVM successfull y invokes the main method	The JVM fails at runtime with NoSuchMethod Error			The JVM fails at runtime with NoSuchMethod Error
What does the AWT stands for ?	Abstract Windowing toolkit	A web toolkit	Application with types	Absolutly wonderfull toolkit			Abstract Windowing toolkit
Which of the following is generated when the source code is successfully compiled ?	Output	Bytecode	Error	None of above			Bytecode
In java , if you do not give a value to a variable before using it ,_____	It will contain a garbage value	It will initialized with zero	Compiler will give an error	None of above			Compiler will give an error

Which among the following is the compulsory section of java program ?	package Statement	import Statement	Documentation section	Class declaration section			Class declaration section
Sharing of common information is achieved by the concept of ?	polymorphism	encapsulation	inheritance	none of above			inheritance
The extension name of a Java source code file is ?	.java	.obj	.class	.exe			.java
The JDK command to compile a class in the file Test.java is	java Test	javac Test.java	javac Test				javac Test.java
_____ is a software that interprets Java bytecode.	Java virtual machine	Java compiler	Java debugger	Java API			Java virtual machine
Which JDK command is correct to run a Java application in ByteCode.class?	java ByteCode	java ByteCode.class	javac ByteCode.java	javac ByteCode			java ByteCode
Java is also known as ..... stage language	One	Two	Three	Four			Two
For interpretation of java program, _____ command is used.	java	javac	javap	none of these			java

What do you mean by javap?	java compiler	java Interpreter	java Disassemble	java debugger			java Disassemble
What is HotJava ?	system software	web browser	java environment	IDE			web browser
Who invented JAVA?	James Gosling	Dennis Ritchie					James Gosling
What was the name of JAVA initially?	Bean	Java Bean	Oak				Oak
Dose Java supports operator overloading?	Yes	No					No
Java supports Pass by reference or Pass by value	Pass by value	Pass by reference					Pass by reference
if(x<y){x=y;y=0;}Is both statement execute in java?	TRUE	FALSE					TRUE
In java Char allocate how many bit?	8 bit	16 bit					16 bit
String contain in which package?	java.util	java.lang					java.lang
one class extend any number of class and interface ?	TRUE	FALSE					FALSE
java is pure object oriented language?	Yes	No					Yes
Is constructor is private or not?	TRUE	FALSE					TRUE

Which is a reserved word in the Java programming language?	method	native	subclasses	array			native
instance variable declare in	class	constructor					class
valid keyword in java	String	Interface					Interface
which class can not be subclass in java	parent class	final class					final class
can we declare abstract static method	Yes	No					No
can we access private class outside the package	Yes	No					No
Thread can be created by how many way	1	2	3	4			2
Runnable is	class	method	Interface				Interface
Which collection class associates values witch keys, and orders the keys according to their natural order	java.util.HashSet	java.util.TreeMap					java.util.TreeMap
In Runnable, many threads share the same object instance	TRUE	FALSE					TRUE
Java beans have no types	TRUE	FALSE					TRUE
The JDBC-ODBC bridge is	single thread	Multithread					Multithread

The Externizable interface extends the serializable interface	TRUE	FALSE					TRUE
Which method executes only once	start()	init()	destroy()				init()
Minimum threads in a program are	1	2	3	4			1
JIT meaning	just in time	java in time					just in time
after compilation of java class the file create is	.class	.doc	.java				.class
constructor has return type	TRUE	FALSE					FALSE
x=x+1 is equivalent to	++x	x++					x++
Inheritance means	subclass extend parent class	subclass implement parent class					subclass extend parent class
Which type of inheritance is not supported by java	multiple	multilevel					multiple
Super is the predefined	keyword	method	keyword and method				keyword and method
All methods of interface are public and abstract	TRUE	FALSE					TRUE
In java, gc() method is available in which package	java.lang package	java.util package					java.lang package
Java intermediate code is known as	Byte code	interpreted cod					Byte code

Interface support multiple extend	TRUE	FALSE					TRUE
can we create the object of abstract class	TRUE	FALSE					FALSE
for declare a constant variable we use keyword	final	volatile					final

UNIT-II

Questions	opt1	opt2	opt3	opt4	opt5	opt6	answer
Which of these keywords is used to define interfaces in Java?	getDta()	GetResponse()	getStream()	interface			interface
Which of these can be used to fully abstract a class from its implementation ?	log()	Interfaces	logHttpd()	logResponse()			Interfaces
Which of these access specifiers can be used for an interface?	findfromCache()	findFromCache()	Public	getFromCache()			Public
Which of these keywords is used by a class to use an interface defined previously?	hits	hitstocache	hits_to_cache	implements			implements

Which of the following is correct way of implementing an interface salary by class manager?	http	httpDecoder	httpConnection	class manager implements salary {}			class manager implements salary {}
Which of the following package stores all the standard java classes?	java	writetoDisk()	writeCache()	writeDiskEntry()			java
Which of these class is superclass of all other classes?	Object	start()	runThread()	startThread()			Object
Which of these method of Object class can generate duplicate copy of the object on which it is called?	Handle()	HandleGet()	clone()	Handleget()			clone()
What is the value of double constant 'E' defined in Math class?	approximately 2.72	Piggy backing	Back packing	Good packing			approximately 2.72
Which of these class contains only floating point functions?	4000 bauds \ sec & 1000 bps	2000 bauds \ sec & 1000 bps	Math	1000 bauds \ sec & 4000 bps			Math
Which of these class encapsulate the run time state of an object or an interface?	Class	Sky	Line of sight	Space			Class
Which of these class have only one field 'TYPE'?	101	Void	105	107			Void

Standard output variable ‘out’ is defined in which class?	Hyper Text Transmission Protocol (HTTP)	Simple Network Management Protocol (SNMP)	System	Simple Mail Transfer Protocol (SMTP)			System
Which of these class can encapsulate an entire executing program?	Class A	Class B	Class C	Process			Process
Which of these class holds a collection of static methods and variables?	Physical	Data-link	Transport	System			System
Which of these is a wrapper for data type int?	Error detection	Error correction	Integer	Slowing down the Communication			Integer
Which of these is a super class of wrappers Long, Character & Integer?	Number	IP	Logical	Physical			Number
Which of these is wrapper for simple data type char?	Class A	Class B	Class C	Character			Character
Which of these keywords is used to define packages in Java?	LogMessage	LogResponse	package	httpdResponse			package
Which of these is a mechanism for naming and visibility control of a class and its content?	httpServer	ServerSockets	MimeHeader	Packages			Packages
Which of these access specifiers can be used for a class so that it’s members can	http	https	Mime	Public			Public

be accessed by a different class in the different package?							
Which of the following is correct way of importing an entire package 'pkg'?	parse()	toString()	getString()	import pkg.*			import pkg.*
Which of the following package stores all the standard java classes?	http	httpDecoder	httpConnection	httpd			java
Which of these can be used to fully abstract a class from its implementation?	string()	toString()	convertString()	Interfaces			Interfaces
Which of these access specifiers can be used for an interface?	port	cache	log	Public			Public
Which of these keywords is used by a class to use an interface defined previously?	port	cache	log	implements			implements
Which of these keywords is used to define packages in Java?	java.io	java.util	java.net	package			package
Which of these is a mechanism for naming and visibility control of a class and its content?	TCIP/IP	DNS	Socket	Packages			Packages
How many ports of TCP/IP are reserved for specific protocols?	10	1024	2048	512			1024
How many bits are in a single IP address?	8	16	32	64			32

Which of these is a full form of DNS?	Data Network Service	Data Name Service	Domain Network Service	Domian Name Service			Domian Name Service
Which of these class is used to encapsulate IP address and DNS?	DatagramPacket	URL	InetAddress	ContentHandler			InetAddress
Which of these stream contains the classes which can work on character stream?	InputStream	OutputStream	Character Stream	All of the mentioned			Character Stream
Which of these class is used to read characters in a file?	FileReader	FileWriter	FileInputStream	InputStreamReader			FileReader
Which of these method of FileReader class is used to read characters from a file?	read()	scanf()	get()	getInteger()			read()
Which of these class can be used to implement input stream that uses a character array as the source?	BufferedReader	FileReader	CharArrayReader	FileArrayReader			CharArrayReader
Which of these is a method to clear all the data present in output buffers?	clear()	flush()	fflush()	close()			flush()
Which of these classes can return more than one character to be returned to input stream?	BufferedReader	Bufferedwriter	PushbachReader	CharArray Reader			PushbachReader
Which of these packages contain classes and interfaces used for input & output operations of a program?	java.util	java.lang	java.io	All of the mentioned			java.io
Which of these class is not a member class of java.io package?	String	StringReader	Writer	File			String

Which of these interface is not a member of java.io package?	DataInput	ObjectInput	ObjectFilter	FileFilter			ObjectFilter
Which of these class is not related to input and output stream in terms of functioning?	File	Writer	InputStream	Reader			File
Which of these is specified by a File object?	a file in disk	directory path	directory in disk	None of the mentioned			directory in disk
Which of these is method for testing whether the specified element is a file or a directory?	IsFile()	isFile()	Isfile()	isfile()			isFile()
Show some networking terminologies given below?	IP Address	Protocol	MAC Address	All mentioned above			All mentioned above
TCP,FTP,Telnet, SMTP,POP etc. are examples of ?	Socket	IP Address	Protocol	MAC Address			Protocol
Which classes are used for connection-oriented socket programming?	Socket	ServerSocket	Both A & B	None of the above			Both A & B
Which class can be used to create a server socket. This object is used to establish communication with the clients?	Socket	ServerSocket	Both A & B	None of the above			ServerSocket
Which methods are commonly used in ServerSocket class?	public OutputStream getOutputStream()	public Socket accept()	public synchronized void close()	None of the above			public Socket accept()

URL is an acronym for?	Uniform Resource Locator	Unified Resource Locator	Uniform Restore Locator	Unified Restore Locator			Uniform Resource Locator
The URLConnection class can be used to read and write data to the specified resource referred by the URL?	TRUE	FALSE					TRUE
The java.net.InetAddress class represents an?	Socket	IP Address	Protocol	MAC Address			IP Address
In InetAddress class which method it returns the host name of the IP Address?	public String getHostName()	public String getHostAddress()	public static InetAddress getLocalHost()	None of the above			public String getHostName()
Which classes are used for connection-less socket programming?	DatagramSocket	DatagramPacket	Both A & B	None of the above			Both A & B
Which class is message that can be sent or received. If you send multiple packet, it may arrive in any order,Moreover, packet delivery is not guaranteed?	DatagramSocket	DatagramPacket	Both A & B	None of the above			DatagramSocket
Which constructor of DatagramSocket class is used that it creates a datagram socket and binds it with the given Port	DatagramSocket(int port)	DatagramSocket(int port, InetAddress address)	Datagram Socket()	None of the above			DatagramSocket(int port, InetAddress address)

Number?							
Which steps occur when establishing a TCP connection between two computers using sockets?	The server instantiates a ServerSocket object, denoting which port number communication is to occur on	The server invokes the accept() method of the ServerSocket class. This method waits until a client connects to the server on the given port	After the server is waiting, a client instantiates a Socket object, specifying the server name and port number to connect to	All of the above			All of the above
The flush() method of PrintStream class flushes any uncleared buffers in memory?	TRUE	FALSE					TRUE
Which method of URL class represents a URL and has complete set of methods to manipulate URL in Java?	java.net.URL	java.net.URLConnection	Both A & B	None of the above			java.net.URL
URL stands for Uniform Resource Locator and represents a resource on the World Wide Web, such as a	TRUE	FALSE					TRUE

Web page or FTP directory?							

UNIT-III

Questions	opt1	opt2	opt3	opt4	opt5	opt 6	answer
RMI allows an object to invoke methods on an object running in another JVM?	TRUE	FALSE					TRUE
RMI uses which objects for the communication with the remote object?	Stub	Skeleton	Both Stub & Skeleton	None of the above			Both Stub & Skeleton
Which is an object, acts as a gateway for the client side, all the outgoing requests are routed through it, and it resides at the client side and represents the remote object?	Stub	Skeleton	Both A & B	None of the above			Stub

When the skeleton receives the incoming request, it does the following tasks?	It reads the parameter for the remote method	It waits for the result	It writes and transmits (marshals) the result to the caller	It reads (unmarshals) the return value or exception	Both It reads the parameter for the remote method & It writes and transmits (marshals) the result to the caller		Both It reads the parameter for the remote method & It writes and transmits (marshals) the result to the caller
In RMI Architecture which layer Intercepts method calls made by the client/redirects these calls to a remote RMI service?	Stub & Skeleton Layer	Application Layer	Remote Reference Layer	Transport Layer			Stub & Skeleton Layer

RMI has which of these protocols implementations?	Java Remote Method Protocol (JRMP)	Internet Inter-ORB Protocol (IIOP)	Jinni Extensible Remote Invocation (JERI)	All mentioned above			All mentioned above
In RMI which layer defines and supports the invocation semantics of the RMI connection, this layer maintains the session during the method call?	The Stub & Skeleton Layer	The Application Layer	The Remote Reference Layer	The Transport Layer			The Remote Reference Layer
RMI uses which protocol on top of TCP/IP (an analogy is HTTP over TCP/IP)?	Java Remote Method Protocol (JRMP)	Internet Inter-ORB Protocol (IIOP)	Jinni Extensible Remote Invocation (JERI)	All mentioned above			Java Remote Method Protocol (JRMP)
Which method of the Naming class (found in java.rmi) is used to update the RMI registry on the server machine?	rebind ()	lookup()	Both A & B	None of the above			rebind ()
1099 is the default port used by RMI Registry?	TRUE	FALSE					TRUE

In Naming class which method specifies name to a remote object?	bind(string name)	rebind(string name)	Both A & B	None of the above			bind(string name)
Abbreviate the term DGC?	Digital Garbage Collection	Distributed Garbage Collection	Distributed Garbage Connection	None of the above			Distributed Garbage Collection
In RMI the objects are passed by Value or Reference?	Objects are passed by value	Objects are passed by Reference	Objects are passed by value and reference	None of the above			Objects are passed by value
The UnicastRemoteObject class provides support for point-to-point active object references using TCP streams?	TRUE	FALSE					TRUE
In a RMI Client Program, what are the exceptions which might have to handled?	RemoteException	NotBoundException	MalFormedURLException	All mentioned above			All mentioned above

Which is a one-way communication only between the client and the server and it is not a reliable and there is no confirmation regarding reaching the message to the destination ?	TCP/IP	UDP	Both A & B	None of the above			UDP
RMI Architecture consists of how many layers?	5	3	4	2			4
An RMI Server is responsible for,	Creating an instance of the remote object	Exporting the remote object	Binding the instance of the remote object to the RMI registry	All mentioned above			All mentioned above
In RMI Distributed object applications need to do?	Locate remote objects	Communicate with remote objects	Load class definitions for objects that are passed around	All mentioned above			All mentioned above

In RMI applications which program obtains a remote reference to one or more remote objects on a server and then invokes methods on them?	Server	Client	Both A & B	None of the above			Client
In RMI program the following example shows the, import java.rmi.*; public interface Adder extends Remote{ public int add(int x,int y)throws RemoteException; }	Create and start the remote application	Create and start the client application	Create the remote interface	Provide the implementation of the remote interface			Create the remote interface

In RMI program the following two steps are used to, Either extend the UnicastRemoteObject class, the exportObject() method of the UnicastRemoteObject class,	Provide the Implementation of the remote interface	Create the remote interface	Create and start the remote application	Compile the implementation class and create the stub and skeleton objects using the rmic tool			Provide the Implementation of the remote interface
Which package is used for Remote Method Invocation (RMI)?	java.lang.rmi	java.lang.reflect	java.applet	java.rmi			java.rmi
Java supports RMI, RMI Stands for?	Random Method Invocation	Remote Method Invocation	Remote Memory Interface	Random Method Invocation			Remote Method Invocation
RMI uses a layered architecture; each of the layers could be enhanced or replaced without affecting the rest of the system?	TRUE	FALSE					TRUE
RMI is a server-side component; It is not required to be deployed	TRUE	FALSE					FALSE

on the server?							
Which is built on the top of socket programming?	EJB	RMI	Both A & B	None of these			RMI
RMI and EJB, provides services to access an object running in another JVM (known as remote object)?	TRUE	FALSE					TRUE
Extensible Markup Language (XML) is much more similar like?	HTML	c++	JavaScript	C			HTML
In the given below which is not a function of XML?	Transport information	Style information	Store information	Structure information			Style information
Which was first most widely used programming interface for accessing relational databases, It offers the ability to	JDBC API	ODBC API	Both A & B	None of the above			ODBC API

connect to almost all databases on almost all platforms.?							
Microsoft has introduced UDA as an umbrella term that covers?	OLE DB	ADO	RDS	All mentioned above			All mentioned above
Which models does JDBC API supports for database access?	Two-tier models	Three-tier models	Both Two-tier models & Three-tier models	None of the above			Both Two-tier models & Three-tier models
In which model a Java applet or application talks directly to the data source?	Two-tier models	Three-tier models	Both Two-tier models & Three-tier models	None of the above			Two-tier models
The JDBC API is what allows access to a data source from a Java middle tier?	TRUE	FALSE					TRUE
In which of the following JDBC a server technology are its	Connection pooling	Distributed transactions	Disconnecte d rowsets	All mentioned above			All mentioned above

support for?							
How many JDBC product components does the Java software provides?	3	2	5	6			3
In the following which JDBC product components does the Java software provides?	the JDBC driver manager	the JDBC driver test suite	the JDBC-ODBC bridge	All mentioned			All mentioned above
Which class has traditionally been the backbone of the JDBC architecture?	the JDBC driver manager	the JDBC driver test suite	the JDBC-ODBC bridge	All mentioned			the JDBC driver manager
Which provides some confidence that JDBC drivers will run your program?	the JDBC-ODBC bridge	the JDBC driver manager	the JDBC driver test suite	None of these			the JDBC driver test suite
JDBC technology-based drivers generally fit into how many categories?	4	3	2	5			4

Which kind of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, IBM DB2, or other DBMS?	JDBC-ODBC bridge plus ODBC driver	Native-API partly-Java driver	JDBC-Net pure Java driver	Native-protocol pure Java driver			Native-API partly-Java driver
In which driver Network connection is indirect that a JDBC client makes to a middleware process that acts as a bridge to the DBMS server?	JDBC-Net	JDBC-ODBC bridge	Native API as basis	Native protocol as basis			JDBC-Net
A leading database connectivity vendor, worked together to produce the?	JDBC-ODBC Bridge	JDBC Driver Test Suite	Both JDBC-ODBC Bridge & JDBC Driver Test Suite	None of these			Both JDBC-ODBC Bridge & JDBC Driver Test Suite
Which list gives a quick way to determine which Connection method is appropriate for creating different types of	createStatement	PrepareStatement	prepareCall	All mentioned			All mentioned

SQL statements ?							
Which method is used for an SQL statement that is executed frequently?	PrepareStatement	prepareCall	createStatement	None of the above			PrepareStatement
The ResultSet.next method is used to move to the next row of the ResultSet, making it the current row?	TRUE	FALSE					TRUE
How many Result sets available with the JDBC 2.0 core API, The following constants, defined in the ResultSet interface, are used to specify these?	2	3	4	5			3

In which the result set generally does not show changes to the underlying database that are made while it is open. The membership, order, and column values of rows are typically fixed when the result set is created?	TYPE_FORWARD_ONLY	TYPE_SCROLL_INSENSITIVE	TYPE_SCROLL_SENSITIVE	ALL MENTIONED ABOVE			TYPE_SCROLL_INSENSITIVE
In concurrency which Indicates a result set that cannot be updated programmatically?	CONCUR_UPDATABLE	CONCUR_READ_ONLY	BOTH CONCUR_UPDATABLE & CONCUR_READ_ONLY	None of these			CONCUR_READ_ONLY
Which methods returns a stream that simply provides the raw bytes from the database without any conversion?	getCharacterStream	getBinaryStream	getAsciiStream	getUnicodeStream			getBinaryStream

Which method is used for retrieving streams of both ASCII and Unicode characters is new in the JDBC 2.0 core API?	getCharacterStream	getBinaryStream	getAsciiStream	getUnicodeStream			getCharacterStream
Drivers that are JDBC Compliant should normally support scrollable result sets, but they are not required to do so?	TRUE	FALSE					TRUE
The intent is for JDBC drivers to implement nonscrollable result sets using the support provided by the underlying database systems?	TRUE	FALSE					FALSE

In order to transfer data between a database and an application written in the Java programming language, the JDBC API provides which of these methods?	Methods on the ResultSet class for retrieving SQL SELECT results as Java types.	Methods on the PreparedStatement class for sending Java types as SQL statement parameters.	Methods on the CallableStatement class for retrieving SQL OUT parameters as Java types.	All mentioned			All mentioned
The JDBC types BINARY, VARBINARY, and LONGVARBINARY are closely related in that VARBINARY represents a?	a small variable-length binary value	a small fixed-length binary value	a large variable-length binary value	Both a small variable-length binary value & a small fixed-length binary value			a small variable-length binary value
In which the JDBC type represents a 64-bit signed integer value between -9223372036854775808 and 9223372036854775807?	SMALLINT	BIGINT	TINYINT	INTEGER			BIGINT

In which the JDBC type represents a "single precision" floating point number that supports seven digits of mantissa?	REAL	DOUBLE	FLOAT	INTEGER			REAL
In Which type may optionally have a custom mapping to a class in the Java programming language,A custom mapping consists of a class that implements the interface SQLData and an entry in a java.util.Map object?	ARRAY	CLOB	DISTINCT	BLOB			DISTINCT
The JDBC API has always supported persistent storage of objects defined in the Java programming	TRUE	FALSE					TRUE

language through the methods getObject and setObject?							
In Advanced JDBC Data types which have defined new data types that are commonly referred to as SQL3 types?	ISO(International Organization for Standardization)	IEC(the International Electrotechnical Commission)	Both ISO(International Organization for Standardization) & IEC(the International Electrotechnical Commission)	None of these			Both ISO(International Organization for Standardization) & IEC(the International Electrotechnical Commission)

UNIT-IV

questions	opt1	opt2	opt3	opt4	opt5	opt6	answer
Which is mandatory in <jsp:useBean /> tag?	id, class	id, type	type, property	type,id			id, class
Which is not a directive?	include	page	export	useBean			export

“request” is instance of which one of the following classes?	Request	HttpRequest	HttpServlet Request	ServletReq uest			HttpServlet Request
Which one is the correct order of phases in JSP life cycle?	Initialization, Cleanup, Compilation, Execution	Initialization, Compilation, Cleanup, Execution	Compilation, Initialization, Execution, Cleanup	Cleanup, Compilatio n, Initializatio n, Execution			Compilatio n, Initializatio n, Execution, Cleanup
Default value of autoFlush attribute is?	TRUE	false					TRUE
Which option is true about session scope?	Objects are accessible only from the page in which they are created	Objects are accessible only from the pages which are in same session	Objects are accessible only from the pages which are processing the same request	Objects are accessible only from the pages which reside in same application			Objects are accessible only from the pages which are in same session

_jspService() method of HttpJspPage class should not be overridden.	TRUE	false					TRUE
Application is instance of which class?	javax.servlet.Application	javax.servlet.HttpContext	javax.servlet.Context	javax.servlet.ServletContext			javax.servlet.ServletContext
Which tag should be used to pass information from JSP to included JSP?	Using <%jsp:page> tag	Using <%jsp:param> tag	Using <%jsp:import> tag	Using <%jsp:useBean> tag			Using <%jsp:page> tag
Which page directive should be used in JSP to generate a PDF page?	contentType	generatePdf	typePDF	contentPDF			contentType
Which one of the following is correct for directive in JSP?	<%@directive%>	<%!directive%>	<%directive%>	<%=directive%>			<%@directive%>

Which of the following action variable is used to include a file in JSP?	jsp:setProperty	jsp:getProperty	jsp:include	jsp:plugin			jsp:include
Which attribute uniquely identification element?	ID	Class	Name	Scope			ID
“out” is implicit object of which class?	javax.servlet.jsp.PrintWriter	javax.servlet.jsp.SessionWriter	javax.servlet.jsp.SessionPrinter	javax.servlet.jsp.JspWriter			javax.servlet.jsp.JspWriter
Which object stores references to the request and response objects?	sessionContext	pageContext	HttpSession	sessionAttribute			pageContext
What temporarily redirects response to the browser?	<jsp:forward>	<%@directive%>	response.sendRedirect(URL)	response.setRedirect(URL)			response.sendRedirect(URL)

Which tag is used to set a value of a JavaBean?	<c:set>	<c:param>	<c:choose>	<c:forward >			<c:set>
Can <!–comment–> and <%–comment–%> be used alternatively in JSP?	TRUE	FALSE					FALSE
Java code is embedded under which tag in JSP?	Declaration	Scriptlet	Expression	Comment			Scriptlet
Which of the following is not a directive in JSP?	page directive	include directive	taglib directive	command directive			command directive
What does foo.getClass().getMethod(“doSomething”, null) return?	doSomething method instance	Method is returned and we can call the method as method.invoke(foo,null);	Class object	Exception is thrown			Method is returned and we can call the method as method.invoke(foo,null);

What does Class.forName("myreflection.Foo").getInstance() return?	An array of Foo objects	class object of Foo	Calls the getInstance() method of Foo class	Foo object			Foo object
How to get the class object of associated class using Reflection?	Class.forName("className")	Class.name("className")	className.getClass()	className.getClassName()			Class.forName("className")
How method can be invoked on unknown object?	obj.getClass().getDeclaredMethod()	obj.getClass().getDeclaredField()	obj.getClass().getMethod()	obj.getClass().getObject()			obj.getClass().getMethod()
How private field can be called using reflection?	getDeclaredFields	getDeclaredMethods	getMethods	getFields			getDeclaredFields
How private method can be called using reflection?	getDeclaredFields	getDeclaredMethods	getMethods	getFields			getDeclaredMethods

What is not the advantage of Reflection?	Examine a class's field and method at runtime	Construct an object for a class at runtime	Examine a class's field at compile time	Examine an object's class at runtime			Examine a class's field at compile time
Which of the following is not a marker interface?	Serializable	Cloneable	Remote	Reader			Reader
Which of the following is used for session migration?	Persisting the session in database	URL rewriting	Create new database connection	Kill session from multiple sessions			Persisting the session in database
Which of the below is not a session tracking method?	URL rewriting	History	Cookies	SSL sessions			History
Which of the following is stored at client side?	URL rewriting	Hidden form fields	SSL sessions	Cookies			Cookies
Which of the following leads to high network traffic?	URL rewriting	Hidden form fields	SSL sessions	Cookies			URL rewriting

Which of the following is not true about session?	All users connect to the same session	All users have same session variable	Default timeout value for session variable is 20 minutes	New session cannot be created for a new user			Default timeout value for session variable is 20 minutes
SessionIDs are stored in cookies.	TRUE	FALSE					TRUE
What is the maximum size of cookie?	4 KB	4 MB	4 bytes	40 KB			4 KB
How can we invalidate a session?	session.discontinue()	session.invalidate()	session.disconnect()	session.falsify()			session.invalidate()
Which method creates unique fields in the HTML which are not shown to the user?	User authentication	URL writing	HTML Hidden field	HTML invisible field			HTML Hidden field

Which object is used by spring for authentication?	ContextHolder	SecurityHolder	Anonymous Holder	SecurityContextHolder			SecurityContextHolder
How constructor can be used for a servlet?	Initialization	Constructor function	Initialization and Constructor function	Setup() method			Initialization and Constructor function
Can servlet class declare constructor with ServletConfig object as an argument?	TRUE	FALSE					FALSE

What is the difference between servlets and applets? i. Servlets execute on Server; Applets execute on browser ii. Servlets have no GUI; Applet has GUI iii. Servlets creates static web pages; Applets creates dynamic web pages iv. Servlets can handle only a single request; Applet can handle multiple requests	i, ii, iii are correct	i, ii are correct	i, iii are correct	i, ii, iii, iv are correct			i, ii are correct
Which of the following code is used to get an attribute in a HTTP Session object in servlets?	session.getAttribute(String name)	session.alterAttribute(String name)	session.updateAttribute(String name)	session.setAttribute(String name)			session.getAttribute(String name)

Which method is used to get three-letter abbreviation for locale's country in servlets?	Request.getISO3Country()	Locale.getISO3Country()	Response.getISO3Country()	Local.retrieveISO3Country()			Request.getISO3Country()
Which of the following code retrieves the body of the request as binary data?	DataInputStream data = new InputStream()	DataInputStream data = response.getInputStream()	DataInputStream data = request.getInputStream()	DataInputStream data = request.fetchInputStream()			DataInputStream data = request.getInputStream()
When destroy() method of a filter is called?	The destroy() method is called only once at the end of the life cycle of a filter	The destroy() method is called after the filter has executed doFilter method	The destroy() method is called only once at the begining of the life cycle of a filter	The destroyer() method is called after the filter has executed			The destroy() method is called only once at the end of the life cycle of a filter

Which of the following is true about servlets?	Servlets execute within the address space of web server	Servlets are platform-independent because they are written in java	Servlets can use the full functionality of the Java class libraries	Servlets execute within the address space of web server, platform independent and uses the functionality of java class libraries			Servlets execute within the address space of web server, platform independent and uses the functionality of java class libraries
How is the dynamic interception of requests and responses to transform the information done?	servlet container	servlet config	servlet context	servlet filter			servlet filter

Which are the session tracking techniques? i. URL rewriting ii. Using session object iii.Using response object iv. Using hidden fields v. Using cookies vi. Using servlet object	i, ii, iii, vi	i, ii, iv, v	i, vi, iii, v	i, ii, iii, v			i, ii, iv, v
How does applet and servlet communicate?	HTTP	HTTPS	FTP	HTTP Tunneling			HTTP Tunneling
In CGI, process starts with each request and will initiate OS level process.	TRUE	FALSE					TRUE

Which class provides system independent server side implementation?	Socket	ServerSocket	Server	ServerReader			ServerSocket
Which of the following is not a implicit object?	request	response	Cookies	session			Cookies
Which of the scripting of JSP not putting content into service method of the converted servlet?	Declarations	Scriptlets	Expressions	None of these			Expressions

The method forward(request, response) will	return back to the same method from where the forward was invoked	not return back to the same method from where the forward was invoked and the web pages navigation continues	both of these	None of these			return back to the same method from where the forward was invoked
The difference between Servlets and JSP is the …….	translation	compilation	syntax	both of these			syntax
Which option is true about session scope?	Objects are accessible only from the page in which they are created	Objects are accessible only from the pages which are in same session	Objects are accessible only from the pages which are processing the same request	bjects are accessible only from the pages which reside in same application			Objects are accessible only from the pages which are processing the same request

UNIT-V

EJB (Enterprise Java Bean) is used to develop which type of applications in java?	Scalable	Robust	Secured	All mentioned above			All mentioned above
Which middleware services are provided by EJB?	Security	Transaction Management	Both A & B	None of the above			Both A & B
In the development of EJB which version is faster because of simplicity and annotations such as @EJB, @Stateless, @Stateful, @ModelDriven, @PreDestroy, @PostConstruct etc.?	EJB 3	EJB 2	EJB 1	None of the above			EJB 3
EJB is like COM provided by Microsoft but, it is different from?	Java Bean	RMI	Web Services	All mentioned above			All mentioned above
EJB is like COM, Abbreviate the term COM?	Component Object Model	Component Oriented Model	Common Object Model	Common Oriented Model			Component Object Model
Which is a server-side component, it is required to be deployed on the server?	EJB	RMI	Both A & B	None of the above			EJB
EJB technology is built on the top of Socket Programming?	TRUE	FALSE					FALSE
In EJB, which must be written in java language?	Bean component	Bean client	Both A & B	None of the above			Both A & B
In EJB,	TRUE	FALSE					TRUE

middleware services are provided by EJB Container automatically?							
Following are the disadvantages of, 1) Requires application server. 2) Requires only java client. 3) Complex to understand and develop ejb applications.	RMI	EJB	Both A & B	None of the above			EJB
The life cycle of session bean is not maintained by the application server (EJB Container)?	TRUE	FALSE					FALSE
How many types of session beans are available in EJB?	2	3	4	5			3
In which session Bean conversational state between multiple method calls is not maintained by the container in case of ?	Stateful Session Bean	Stateless Session Bean	Singleton Session Bean	None of the above			Stateless Session Bean
In which session bean maintain their state between client invocations but are not required to maintain their state across server crashes or shutdowns?	Stateful Session Bean	Stateless Session Bean	Singleton Session Bean	None of the above			Singleton Session Bean
In which type an instances retain no data or conversational state for a	Message-Driven Bean	Session Bean	Entity Bean	None of the above			Message-Driven Bean

specific client?							
JMS is also known as a messaging service?	TRUE	FALSE					TRUE
Abbreviate the term JSM?	Java Message Service	Java Monitor Service	Java Message Session	Java Monitor Session			Java Message Service
JMS is mainly used to send and receive message from one application to another?	TRUE	FALSE					TRUE
In the advantage of JMS which receive the message, client is not required to send request, Message will arrive automatically to the client?	Reliable	Asynchronous	Both A & B	None of the above			Asynchronous
In which model of message domain is delivered to one receiver only, where Queue is used as a message oriented middleware (MOM)?	Point-to-Point Model	Publisher/Subscriber Model	Both A & B	None of the above			Point-to-Point Model
A message driven bean is like statefull session bean that encapsulates the business logic and doesn't maintain state?	TRUE	FALSE					FALSE
In which component Entity bean represents the	Server-side component	Client-side component	server and client side component	None of the above			Server-side component

persistent data stored in the database?							
In EJB these two types of entity beans are bean managed persistence and container managed persistence are used in?	EJB 2.x	EJB 3.x	Both A & B	None of the above			EJB 2.x
The ASP and JSP technologies are quite similar in the way they support the creation of Dynamic pages, using HTML templates, scripting code and components for business logic?	TRUE	FALSE					FALSE
JSP's provide better facilities for separation of page code and template data by mean of Java beans, EJBs and custom tag libraries?	TRUE	FALSE					TRUE
The authentication mechanism in the servlet specification uses a technique called?	Role Based Authenticati on	Form Based Authentication	Both A & B	None of the above			Both A & B
In JSP which is an exception that is typically a user error or a problem that	Checked exceptions	Runtime exceptions	Errors	None of the above			Runtime exception s

cannot be foreseen by the programmer?							
In which attribute specifies a JSP page that should process any exceptions thrown but not caught in the current page?	The ErrorPage Attribute	The IsErrorPage Attribute	Both A & B	None of the above			Both A & B
Which is the Microsoft solution for providing dynamic Web content?	ASP	JSP	Both A & B	None of the above			None of the above
JSPs eventually are compiled into Java servlets, you can do as much with JSPs as you can do with Java servlets?	TRUE	FALSE					FALSE
Which Error Handling in Java handles runtime errors with exceptions, If an exception is not caught in your JSP or Servlet,Resin will use a special error page to send results back to the browser,Resin uses a default error page unless you explicitly provide an error page yourself?	Client Request Time Processing Errors	Compilation Time Processing Errors	JSP Translation Time Processing Errors	None of the above			None of the above
In which Architecture of a	Model1 Architectur	Model2 Architecture	Both A & B	None of the above			None of the above

JSP Application JSP plays a key role and it is responsible for processing the request made by client?	e						
Seperation of business logic from JSP this is the advantage of?	Custom Tags in JSP	JSP Standard Tag Library	Both A & B	None of the above			Both A & B
Model View Controller in JSP which represents the state of the application i.e. data. It can also have business logic?	Model	View	Controller	None of the above			None of the above
In JSP action tags Which are used for developing web application with Java Bean?	jsp:useBean	jsp:setProperty	jsp:getProperty	Both B & C			Both B & C
A bean encapsulates many objects into one object, so we can access this object from multiple places?	TRUE	FALSE					TRUE
In JSP Action tags which is used to include the content of another resource it may be jsp, html or servlet?	jsp:include	jsp:forward	jsp:plugin	None of the above			None of the above
In JSP Action tags which tags are used for bean development?	jsp:useBean	jsp:setPoperty	jsp:getProperty	All mentioned above			All mentioned above

In JSP how many ways are there to perform exception handling?	3	2	4	5			5
The Jsp include directive is used to include the contents of any resource it may be?	jsp file	html file	text file	All mentioned above			All mentioned above
In JSP page directive which attribute defines the MIME(Multipurpose Internet Mail Extension) type of the HTTP response?	import	Content Type	Extends	Info			Extends
How many jsp implicit objects are there and these objects are created by the web container that are available to all the jsp pages?	8	9	10	7			10
Which tag is used to execute java source code in JSP?	Declaration Tag	Scriptlet tag	Expression tag	None of the above			None of the above
The javax.servlet.jsp package has two interfaces find in the following?	JspPage	HttpJspPage	JspWriter	Both A & B			Both A & B
In the following which packages does a JSP API consists of?	javax.servlet.jsp	java.servlet	javax.servlet.jsp.tagext	Both A & C			Both A & C
In which technology, we mix our business logic with the presentation	Servlet	JSP	Both A & B	None of the above			Both A & B

logic?							
These are the advantages of which technology, Extension to Servlet Easy to maintain; No need to recompile and redeploy Fast Development Less code?	JSP	Servlet	Both A & B	None of the above			Both A & B
A JSP page consists of which tags?	HTML tags	JSP tags	Both A & B	None of the above			Both A & B
In JSP which can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.?	vs.Static HTML	vs.Server-Side Includes	vs.Pure Servlets	Vs.JavaScript			Vs.JavaScript
JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI)?	TRUE	FALSE					TRUE
A session bean represents a multiple clients inside the Application Server?	TRUE	FALSE					TRUE
In Enterprise Beans to accommodate a	Transactions must ensure	The application must be scalable	The application will have a variety of clients	All mentioned above			All mentioned above

growing number of users, you may need to distribute an application's components across multiple machines?	data integrity						
In which an EJB container must provide an implementation of Java Naming and Directory Interface (JNDI) API to provide naming services for EJB clients and components?	Transaction support	Persistence support	Naming support	All mentioned above			Naming support
The EJB container provide services to EJB components they are?	Transaction support	Persistence support	Naming support	All mentioned above			All mentioned above
To run EJB application, you need an application server (EJB Container) such as Jboss, Glassfish, Weblogic, WebSphere etc. It performs which of the following?	Life cycle management	transaction management	object pooling	All mentioned above			All mentioned above
EJB is a specification for J2EE server, not a product; Java beans may be a graphical component in IDE?	TRUE	FALSE					TRUE
In EJB	Bean-	Container-	Both A & B	None of			Bean-

transaction management which case of a session bean obtains the UserTransaction object via the EJBContext using the getUserTransaction() method?	managed transactions	managed transactions		the above			managed transactions
Which represents persistent global data from the database?	Entity Bean	Session Bean	Both A & B	None of the above			Entity Bean
EJB QL is a Query Language provided for navigation across a network of enterprise beans and dependent objects defined by means of container managed persistence?	TRUE	FALSE					TRUE
EJB is a specification for J2EE server, not a product; Java beans may be a graphical component in IDE?	TRUE	FALSE					TRUE
EJB 3.0 provides option to define database entity relationships/mappings like ?	One to One	One to Many	Many to One	All mentioned above			All mentioned above
Entity beans differ from session beans in several ways they are?	persistent	Allow shared access	Have primary keys	All mentioned above			All mentioned above

In which type of EJB it encapsulates the state that can be persisted in the database,it is deprecated,Now , it is replaced with JPA (Java Persistent API)?	Entity Bean	Message Driven Bean	Session Bean	None of the above			Entity Bean