Geethanjali College of Engineering and Technology

Cheeryal (V), Keesara (M), Ranga Reddy District – 501 301 (T.S)

WEB TECHNOLOGIES

COURSE FILE

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

(2015-2016)

Faculty in charge
S.Ramanjaneyulu & P Swathi

HOD-CSE
Dr.S. Nagender Kumar

Contents

S.No	Topic	Page. No.
1	Cover Page	1
2	Syllabus copy	4
3	Vision of the Department	5
4	Mission of the Department	5
5	PEOs and POs	5
6	Course objectives and outcomes	7
7	Course mapping with POs	8
8	Brief notes on the importance of the course and how it fits into the curriculum	9
9	Prerequisites if any	9
10	Instructional Learning Outcomes	9
11	Class Time Table	12
12	Individual Time Table	14
13	Lecture schedule with methodology being used/adopted	14
14	Detailed notes	20
15	Additional topics	70
16	University Question papers of previous years	71
17	Question Bank	75
18	Assignment Questions	78
19	Unit wise Quiz Questions and long answer questions	80
20	Tutorial problems	91
21	Known gaps ,if any and inclusion of the same in lecture schedule	91
22	Discussion topics , if any	91
23	References, Journals, websites and E-links if any	92
24	Quality Measurement Sheets	94
A	Course End Survey	94
B	Teaching Evaluation	
25	Student List	93
26	Group-Wise students list for discussion topic	96

Course coordinator **Program Coordinator** **HOD**

Geethanjali College of Engineering and Technology

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Name of the Subject/Lab Course): Web Technologies

(JNTU CODE :56032 **Programme : UG**

Branch: CSE	**Version No:**
Year: III	**Document Number :**
Semester: II	**No. of Pages:**

Classification status (Unrestricted/Restricted) :

Distribution List:

Prepared by :	**Updated by :**
1) Name : S.RAMANJANEYULU	**1) Name : P SWATHI**
2) Sign :	**2) Sign :**
3) Design : Asst. Prof	**3) Design : Asst. Prof**
4) Date :	**4) Date :**

Verified by :	***For Q.C only**
1) Name :	**1)Name :**
2) Sign :	**2) Sign :**
3) Design :	**3) Design :**
4) Date :	**4) Date :**

Approved by (HOD) :

1) Name:

2) Sign :

3) Date :

2. Syllabus:

UNIT-I:

HTML Common tags- List, Tables, images, forms, Frames; Cascading Style sheets;

UNIT-II:

Introduction to Java Scripts, Objects in Java Script, Dynamic HTML with Java Script

UNIT-III:

XML: Document type definition, XML Schemas, Document Object model, Presenting XML, Using XML Processors: DOM and SAX

UNIT-IV:

Java Beans: Introduction to Java Beans, Advantages of Java Beans, BDK Introspection, Using Bound properties, Bean Info Interface, Constrained properties, Persistence, Customizes, Java Beans API, Introduction to EJB's

UNIT-V:

Web Servers and Servlets: Tomcat web server, Introduction to Servelets: Lifecycle of a Serverlet, JSDK, The Servelet API, The javax.servelet Package, Reading Servelet parameters, Reading Initialization parameters. The javax.servelet HTTP package, Handling Http Request & Responses, Using Cookies-Session Tracking, Security Issues,

UNIT-VI:

Introduction to JSP: The Problem with Servelet. The Anatomy of a JSP Page, JSP Processing. JSP Application Design with MVC Setting Up and JSP Environment: Installing the Java Software Development Kit, Tomcat Server & Testing Tomcat

UNIT-VII:

JSP Application Development: Generating Dynamic Content, Using Scripting Elements Implicit JSP Objects, Conditional Processing – Displaying Values Using an Expression to Set an Attribute, Declaring Variables and Methods Error Handling and Debugging Sharing Data Between JSP pages, Requests, and Users Passing Control and Date between Pages – Sharing Session and Application Data – Memory Usage Considerations

UNIT VIII:

Database Access : Database Programming using JDBC, Studying Javax.sql.* package,Accessing a Database from a JSP Page, Application – Specific Database Actions,Deploying JAVA Beans in a JSP Page, Introduction to struts framework..

TEXT BOOKS:

1. Web Programming, building internet applications, Chris Bates 2nd edition,WILEY Dreamtech (UNIT s 1,2 ,3)
2. The complete Reference Java 2 Fifth Edition by Patrick Naughton and Herbert Schildt. TMH (Chapters: 25) (UNIT 4)
3. Java Server Pages –Hans Bergsten, SPD O'Reilly (UNITs 5,6,7,8)

REFERENCE BOOKS:

1. Programming world wide web-Sebesta,Pearson
2. Core SERVLETS ANDJAVASERVER PAGES VOLUME 1: CORE TECHNOLOGIES By Marty Hall and Larry Brown Pearson
3. Internet and World Wide Web – How to program by Dietel and Nieto PHI/Pearson Education Asia.

4. Jakarta Struts Cookbook , Bill Siggelkow, S P D O'Reilly for chap 8.

3. Vision of the Department

To produce globally competent and socially responsible computer science engineers contributing to the advancement of engineering and technology which involves creativity and innovation by providing excellent learning environment with world class facilities.

4. Mission of the Department

1. To be a center of excellence in instruction, innovation in research and scholarship, and service to the stake holders, the profession, and the public.

2. To prepare graduates to enter a rapidly changing field as a competent computer science engineer.

3. To prepare graduate capable in all phases of software development, possess a firm understanding of hardware technologies, have the strong mathematical background necessary for scientific computing, and be sufficiently well versed in general theory to allow growth within the discipline as it advances.

4. To prepare graduates to assume leadership roles by possessing good communication skills, the ability to work effectively as team members, and an appreciation for their social and ethical responsibility in a global setting.

5. PEOs & Pos

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in industry and / or to pursue postgraduate studies with an appreciation for lifelong learning.

2. To provide graduates with analytical and problem solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.

3. To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting-edge technologies of multi-disciplinary nature for societal development.

PROGRAM OUTCOMES (PO)

1. An ability to apply knowledge of mathematics, science and engineering to develop and analyze computing systems.
2. an ability to analyze a problem and identify and define the computing requirements appropriate for its solution under given constraints.
3. An ability to perform experiments to analyze and interpret data for different applications.
4. An ability to design, implement and evaluate computer-based systems, processes, components or programs to meet desired needs within realistic constraints of time and space.
5. An ability to use current techniques, skills and modern engineering tools necessary to practice as a CSE professional.
6. An ability to recognize the importance of professional, ethical, legal, security and social issues and addressing these issues as a professional.
7. An ability to analyze the local and global impact of systems /processes /applications /technologies on individuals, organizations, society and environment.
8. An ability to function in multidisciplinary teams.
9. An ability to communicate effectively with a range of audiences.
10. Demonstrate knowledge and understanding of the engineering, management and economic principles and apply them to manage projects as a member and leader in a team.
11. A recognition of the need for and an ability to engage in life-long learning and continuing professional development
12. Knowledge of contemporary issues.
13. An ability to apply design and development principles in producing software systems of varying complexity using various project management tools.
14. An ability to identify, formulate and solve innovative engineering problems.

6. Course Objectives & Outcomes

Course Objectives

This course demonstrates an in-depth understanding of the tools and Web technologies necessary for business application design and development. The course covers client side scripting like HTML, JavaScript and server side scripting like servlets, JSPs and also XML and web servers and database interfacing.

Course Outcomes

After Completion of this course Students will be able to

- ✓ Outline the history of the web, and technologies that makes the web pages and publishing them.
- ✓ Make the web pages more dynamic and interactive.
- ✓ Design to create structure of web page, to store the data in web document, and transport information through web.
- ✓ Design to be reusable the software components in a variety of different environments.
- ✓ Install Tomcat Server and execution of programs on server side.
- ✓ Identify the problems in Servlets and overcome those using Java Server Pages also develop JSP applications with Model View Control architecture.
- ✓ Establish the Connection between Java Application and database to insert, retrieve and modify the data in tables.

7. Course Mapping with PO

Course	PEOS	POs
WT	PEO1,PEO2,PEO3	PO1,PO2,PO3,PO4,PO5,PO6,PO7,PO8,PO9,PO11,PO12, PO13,PO14

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
WEB TECHNOLOGIES														
CO316.1 Outline the history of the web, and technologies that makes the web pages and publishing them.	**H**	**H**	**H**	**H**	**H**	**H**	**M**	**M**	**L**	**H**	**H**		**H**	**M**
CO316.2 Make the web pages more dynamic and interactive.	**H**	**H**	**H**	**H**	**H**	**H**	**M**	**M**	**L**	**H**	**H**		**H**	**H**
CO316.3 Design to create structure of web page, to store the data in web document, and transport information through web.	**H**	**H**	**H**	**M**	**M**	**L**		**H**		**M**	**H**	**L**	**H**	**M**
CO316.4 Design to be reusable the software components in a variety of different environments.	**H**	**H**	**H**	**H**	**H**	**L**	**H**	**H**					**M**	**H**
CO316.5 Install Tomcat Server and execution of programs on server side.	**H**	**H**	**H**	**H**	**H**		**H**	**M**		**L**	**H**	**H**	**H**	**H**
CO316.6 Identify the problems in Servlets and overcome those using Java Server Pages also develop JSP applications with Model View Control architecture.	**H**	**H**	**H**	**H**	**M**		**L**	**H**		**M**	**M**	**H**	**M**	**H**
CO316.7 Establish the Connection between Java Application and database to insert, retrieve and modify the data in tables	**H**	**H**	**H**	**H**	**H**	**H**	**H**	**H**		**H**	**H**	**H**	**H**	**M**

8. Brief notes on the importance of the course and how it fits into the curriculum

The study of Web Technologies course demonstrates an in-depth understanding of the tools and Web technologies necessary for business application design and development. The course covers client side scripting like HTML, JavaScript and server side scripting like servlets, JSPs and also XML and web servers and database interfacing. By studying this course, students will develop their own paths in the software industry.

9. Prerequisites if any

- Good programming skills and debugging skills.
- Good knowledge in Java Programming

10. Instructional Learning Outcomes

At the end of the unit Student will be able to

Unit -1

- Outline the origin of markup languages.
- Develop basic web page/s.
- Create a link from one webpage to another.
- Apply image insertion into web page/s and also make hyperlinks to images.
- Use different styles to the page elements.
- Create, modify and format the contents of web page with CSS.

Unit -2

- Create dynamic interactive Web pages using JavaScript.
- Apply basic control of elements with JavaScript.
- Use Java Script to validate form entries.

- Change the appearance of web pages.
- Create HTML that can change even after a page has been loaded into a browser.

Unit -3

- Outline the evolution, theoretical context, and application of XML.
- Develop the basic programming with XML and publishing models.
- Apply practical experience with XML, DTDs, schemas, XSLT and XSL.
- Recognize the relationship of XML and metadata, and the role of XML in libraries.

Unit -4

- Outline the fundamental principles and methodologies of constructing component software including component standards, architectures, interface, implementations, and integrations.
- Design components to solve a problem, and evaluate alternatives.
- Utilize effectively the software component development tools to develop software components such as Java Beans.

Unit -5

- Analyze the role of Java Servlets in Java 2 Enterprise Edition architecture and as the best Java solution to HTTP application development.
- Use request and response objects provided to a servlet to read parameters and to produce an HTML response.

- Develop interactive web applications using HTML forms and servlets.
- Demonstrate the complex conversation with HTTP clients using session attributes.

Unit -6 & 7

- Explain the JSP technology, its features and advantages.
- Discuss the architecture of web-based systems.
- Describe Web development process and varies server-side technologies.
- Develop JSP Applications with JSP tags, JSP scriptlets and java beans.
- Distinguish JSP Application Models.
- Develop JSP applications implementing Session management and Data base Connectivity.

Unit -8

- Outline the fundamentals of JDBC and its importance, uses, strengths and weaknesses.
- Explain the JDBC architecture and required components.
- Manage normal and abnormal interactions with databases using JDBC.
- Evaluate the process results using scrollable result sets and row sets.

11. Class Time Tables III A & C sec

12. Individual Time Table

13. Lecture schedule with methodology being used/adopted

Geethanjali College of Engineering & Technology

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.Tech III year II sem CSE

Sub: **Web Technologies**

Faculty Name : **S.Ramanjaneyulu**

Lesson & Scheduled Plan

S.No.	Unit No.	Date (s)	Topics to be covered	No. of classes
1	I		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			Frames	1
5			Cascading Style sheets	2
6			Examples for CSS HTML tags	2
7			Tutorial Class	1
8			Assignment Test – 1	1
9	II		Introduction to Java Script	1
10			Objects in Java Script	2
11			Dynamic HTML with Java Script	2
12			Programs on Java Script	2
13			Revise University Question papers	1

14			Assignment Test – 2	1
15	**III**		Introduction to XML	1
16			Document type definition	1
17			XML Schemas	1
18			Document Object model	1
19			Presenting XML	1
20			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	**IV**		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			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			Declaring Variables and Methods Error Handling and Debugging	1
55	**VII**		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			Database Programming using JDBC	1
62	**VIII**		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

14. Detailed 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)

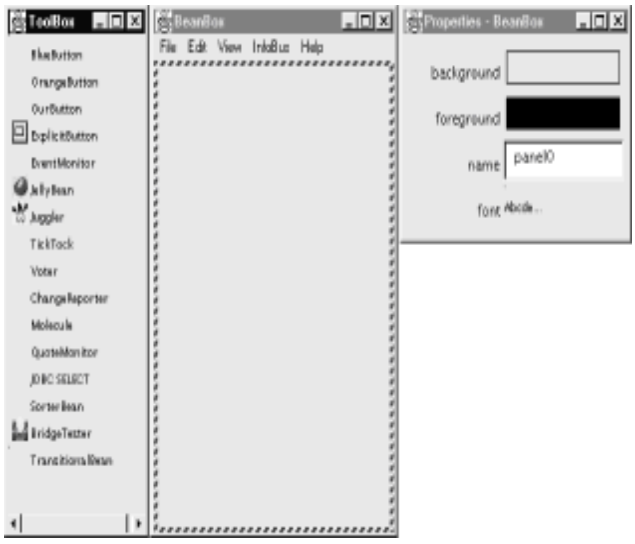
The Bean Developer Kit (BDK), available from the JavaSoft site, is a simple example of a tool that enables you to create, configure, and connect a set of Beans. There is also a set of sample Beans with their source code. This section provides step-by-step instructions for installing and using this tool.

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.

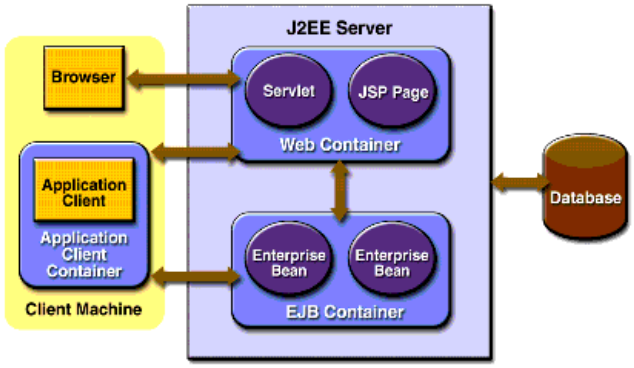
VetoableChangeSupport
Beans that support constrained properties can use this class to notify VetoableChangeListener objects.

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 :

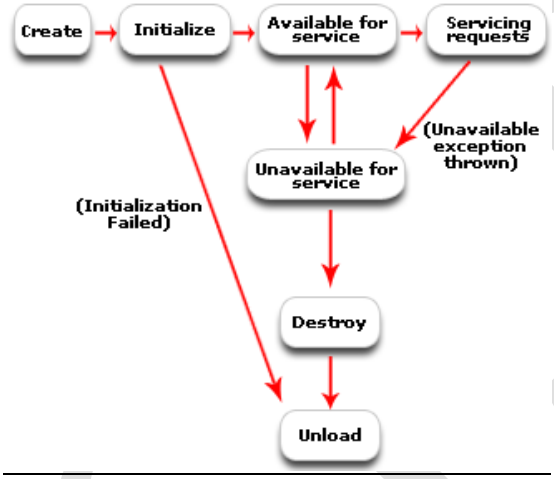
Introduction to servlets
Lifecycle of a servlet
JSDK server
Servlet API
Session tracking
Security issues

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.

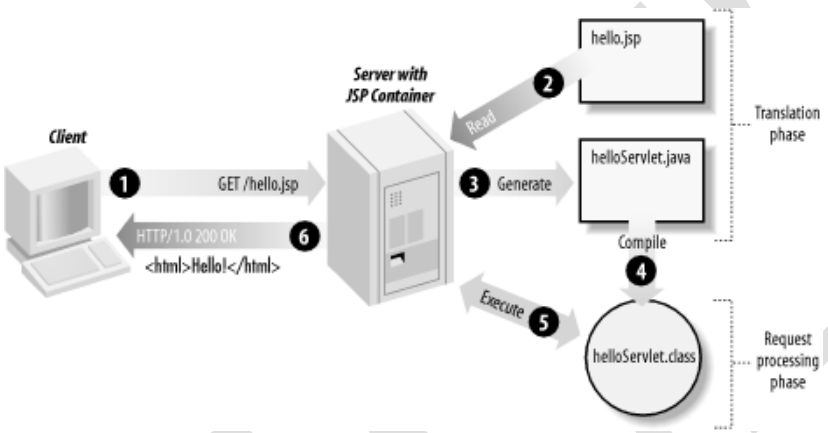
Converting the JSP page to a servlet and compiling the servlet form the *translation phase*.

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.

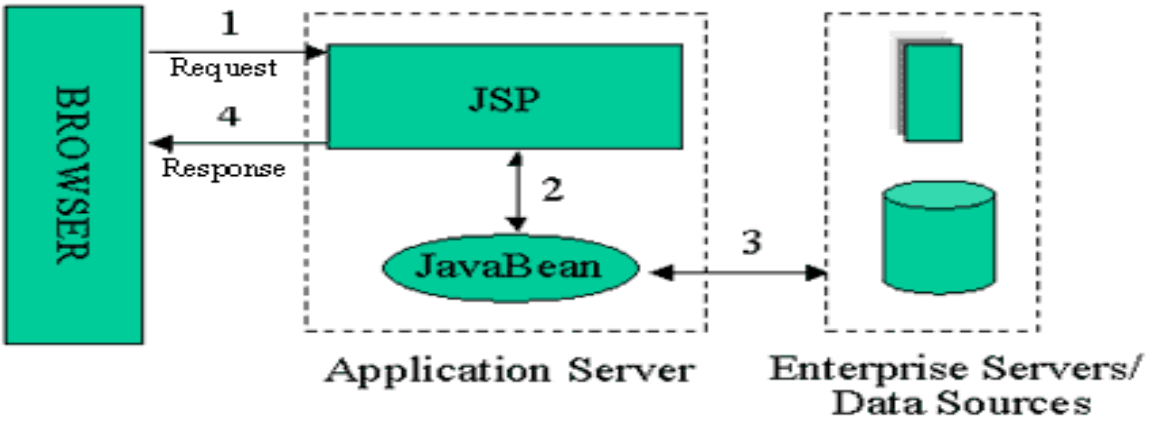
In a server application, we commonly classify the parts of the application as business logic, presentation, and request processing.

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 ... %>`	Includes a file during the translation phase
`<%@ taglib ... %>`	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

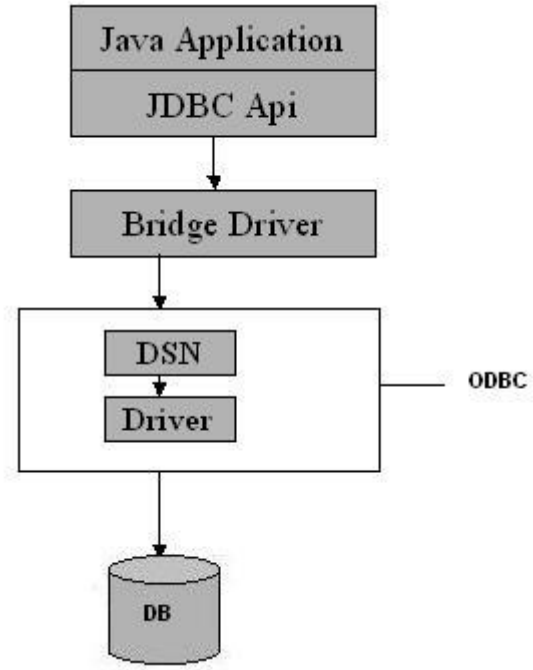
JDBC drivers are divided into four types or levels. The **different types of jdbc drivers** are:

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.

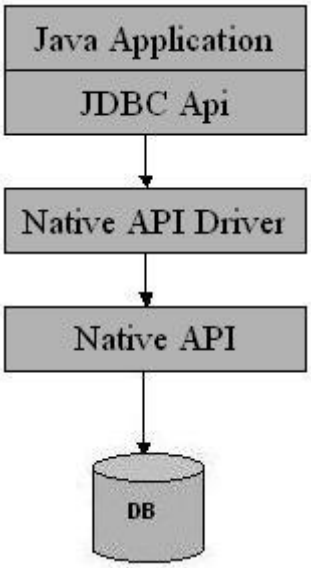
Disadvantages

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.

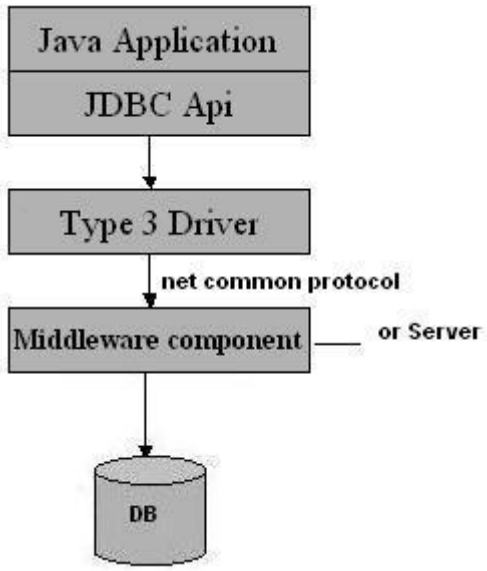
Disadvantage

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.

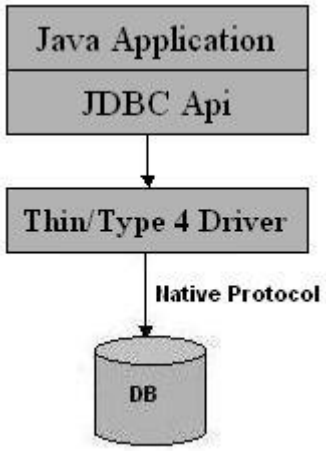
Disadvantage

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).

15.JNTU Syllabus with additional topics :

1. **Session tracking using Hidden form fields.**
2. **Struts framework**

16.University Question papers of previous years

Code No: 09A60505	R09	SET-1

B. Tech III Year II Semester Examinations, April/May-2012
WEB TECHNOLOGIES
(Common tot CSE, IT)

Time: 3 hours **Max. Marks: 75**

Answer any five questions
All questions carry equal marks
---

1. a) Explain the purpose of cascading style sheets.
   b) Design and create the page(s) for accepting the values of name and marks in a table then displays them in the descending order of the marks. [15]

2. With an example explain how parameters are passed to functions in JavaScript. [15]

3. a) What is Document Object Model (DOM)? Explain the DOM levels.
   b) Explain the working of XSL. [15]

4. a) Describe the different types of properties used in Java Beans with an example.
   b) Explain about Introspection. [15]

5. Briefly explain about Tomcat web server. [15]

6. a) What are the limitations of Servlets? How JSP over comes these problems?
   b) Explain the Directive and Scripting elements of JSP. [15]

7. Explain error handling and debugging in JSP pages. [15]

8. a) Explain the process of accessing a database from a JSP page.
   b) Explain struts framework. [15]

Code No: 09A60505	R09	SET-2

B. Tech III Year II Semester Examinations, April/May-2012
WEB TECHNOLOGIES
(Common tot CSE, IT)

Time: 3 hours **Max. Marks: 75**

Answer any five questions
All questions carry equal marks

---

1. Explain the following input components in HTML forms with proper syntax of the corresponding HTML tags.
   a) Text Input
   b) Selectable list with multiple selection option
   c) Radio Buttons. [15]

2. a) Explain the characteristics of DHTML.
   b) Write a JavaScript program to convert distance in kilometers, miles to meters or inches. [15]

3. a) What is XML? How is XML useful? How is XML compatible with others?
   b) Explain DTD sequences. [15]

4. Explain the Enterprise Java Beans technology and its importance on developing the sever-side architecture. [15]

5. a) Explain the life cycle of a servlet.
   b) Write a session tracker that tracks the number of accesses and last access data of a particular web page. [15]

6. a) Explain JSP processing.
   b) How do you test Tomcat server? Explain. [15]

7. a) Describe various steps that are needed for accessing a database from a JSP page.
   b) Discuss about memory usage considerations in JSP application development. [15]

8. Explain the deploying of JAVA Beans in a JSP page. [15]

Code No: 09A60505 R09

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

B. Tech III Year II Semester Examinations, May – 2013

Web Technologies

(Common to CSE, IT)

Time: 3 hours **Max. Marks: 75**

Answer any five questions

All questions carry equal marks

---

1. List and explain various HTML tags with examples and usage. [15]

2. Give a good account of Java Scripts and objects in Java Script. [15]

3. Explain XML schemas and Document Object Model. Give a short note on XML preprocessors. [15]

4. Write Notes on:
   a) BDK introspection
   b) Java Beans API
   c) EJB'S [15]

5.a) Explain the life cycle of a servelet.
   b) Explain the handling of HTTP Requests and Responses.
   c) Explain cookies-session tracking. [15]

6.a) Explain the problems with servelet.
   b) Explain JSP application Design with MVC through an illustrative example. [15]

7. Explain in detail the JSP application development touching upon all the relevant issues and with examples. [15]

8. Write notes on:
   a) Struts Frame work
   b) Data Base Programming with JDBC
   c) Deployment of JAVA beans in a JSP page. [15]

Code No: 09A60505 R09

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

B. Tech III Year II Semester Examinations, December - 2012

WEB TECHNOLOGIES

(Common to CSE, IT)

Time: 3 hours **Max. Marks: 75**

Answer any five questions

All questions carry equal marks

---

1. Create a HTML document that includes 2 rows of frames with 2 frames in each row. The two left frames must occupy 25% of the width of the display. The bottom 2 frames must occupy 40% of the height of the display. The top left frame must display the name of your mother and all of her siblings (min2). The bottom left frame must display the name of your father and all of his siblings (min2). Each name in the left frames must be a link to a document that is displayed in the right frame when the link is selected. The documents in the right frames are short descriptions of the people. [15]

2.a) What object oriented features are supported by Java Script? Explain with examples.

b) Write about the concept, syntax, typical use of OBJECTS in Java Script. [15]

3.a) What is DTD? Explain its purpose? What are the limitations of DTD?

b) "XML Schemas depends on XML syntaxes for their Documentation", explain the feature and its advantages. [15]

4.a) What is a Java Bean? What are the advantages of Java Beans? Illustrate with examples.

b) Explain with example how to develop a simple Bean and connect it to other components using BDK. [15]

5.a) Explain the working of Tomcat web server.

b) Explain javax.servelet.* package. [15]

6.a) What are the problems with Servelet? Explain the anatomy of JSP page.

b) Describe the JSP environment. [15]

7.a) Explain the error handling and debugging in JSP pages.

b) What are sessions? How do you enable and disable sessions using JSP? [15]

8.a) Explain the struts framework.

b) Explain how Java beans are deployed in a JSP page. [15]

--ooOoo--

17. Question Bank

Unit - 1

1. Explain the purpose of cascading style sheets.
2. Design and create the page(s) for accepting the values of name and marks in a table then displays them in the descending order of the marks.
3. Explain the following input components in HTML forms with proper syntax of the corresponding HTML tags. Text Input, Selectable list with multiple selection option, Radio Buttons
4. Write HTML code to create a frame with a table contents on the left side of the window, and have each entry in the table of contents. Use internal linking to scroll down the document frame to the appropriate subsection.
5. How can we create nested lists? Explain with an example?
6. Explain the syntaxes of anchor tags?
7. How can we create forms in HTML? Explain with examples?
8. What is a frame? How can we create nested frames?
9. Explain different forms of selectors?
10. What is List? Explain different types of lists used in HTML?
11. What is CSS? Explain different mechanisms used to apply CSS to HTML pages?

Unit -2

1. With an example explain how parameters are passed to functions in JavaScript.
2. Explain the characteristics of DHTML.
3. Write a JavaScript program to convert distance in kilometers, miles to meters or inches
4. Write a java script to verify a phone number, email-id and date formats.
5. Compare and contrast HTML and DHTML with suitable examples
6. Explain the need for scripting languages in web programming.
7. Explain the features of Java Script.
8. What is JavaScript ? write the features of JavaScript?
9. Write the differences between HTML and DHTML
10. Explain any three objects of JavaScript
11. What is form validation? Explain with example?
12. What is an event ? how can we handle events in JavaScript?
13. What is exception? Explain different ways of handling exceptions in JavaScript?

Unit – 3

1. What is an XML schema? With an example explain the working of XML schema.
2. What is a document type definition (DTD)? How do you create DTD?

3. Explain DTD sequences.
4. What is XML? How is XML useful? How is XML compatible with others?
5. What is Document Object Model (DOM)? Explain the DOM levels.
6. Explain the working of XSL.
7. What is XML? Write the differences between XML and HTML
8. What is DTD? Explain the building blocks of DTD?
9. What is XML Schema? Explain the advantages of Schema over DTD?
10. Explain SAX parser?
11. Explain DOM parser? What are the difference between SAX and DOM

Unit – 4

1. Describe the different types of properties used in Java Beans with an example.
2. Describe the bean info interface.
3. Explain the advantages and disadvantages of Java Beans.
4. How are EJBs related to simple java beans? What are the difference between them?
5. Describe BDK Introspection.
6. Explain the Enterprise Java Beans technology and its importance on developing the server – side architecture.
7. What is javaBean ? write the advantages of Java Beans?
8. What is introspection? Explain in detail?
9. What is BDK? Explain the steps involved in BDK tool?
10. Explain persistence and customizers?
11. Explain about EJB Architecture.

Unit – 5

1. Write a session tracker that tracks the number of accesses and last access data of a particular web page.
2. What are the security issues related to servlets.
3. Explain how Http POST request is processed using Servlets
4. How cookies are used for session tracking?
5. Briefly explain about Tomcat web server.
6. What is servlet? Explain life cycle of a servlet?
7. What are the advantages of servlets over CGI
8. What is session tracking? Explain different mechanisms of session tracking?
9. Explain classes and interfaces of javax.servlet package
10. Explain classes and interfaces of javax.servlet.http package
11. What are the security issues in servlets?
12. What's the difference between servlets and applets?
13. What is the difference between Difference between doGet() and doPost()?
14.

Unit – 6

1. What are the problems with servlet?
2. Explain testing of tomcat.
3. Explain the anatomy of JSP page.
4. Give an overview of java software development kit.
5. Explain JSP processing.
6. Write the advantages of JSP over servlets
7. What is MVC explain with neat diagram?
8. What are the difference between JSP and ASP?

Unit – 7

1. Describe various steps that are needed for accessing a database from a JSP page.
2. Discuss about memory usage considerations in JSP application development.
3. Explain the procedure of how JSP is used to generate dynamic data using an example.
4. Discuss about implicit objects in JSP.
5. Explain error handling and debugging in JSP pages.
6. Explain different JSP elements?
7. Explain conditional processing with examples?
8. How can we share data between JSP pages?
9. How can we implement session tracking in JSP pages?
10. What are the memory usage considerations

Unit – 8

1. Explain the process of accessing a database from a JSP page
2. Explain Struts framework.
3. Explain Javax.sql.* package.
4. Explain the deploying of JAVA Beans in a JSP page.
5. What is JDBC? Explain the JDBC architecture.
6. Write a JDBC program how to insert and updates records into database.
7. Explain different types of JDBC drivers with suitable examples.
8. Describe the JDBC packages.
9. Compare between JDBC servlets and JDBC JSP with example.

18. ASSIGNMENT QUESTIONS

Unit-I

1. How can we create nested lists? Explain with an example?
2. Explain the syntaxes of anchor tags?
3. How can we create forms in HTML? Explain with examples?
4. What is a frame? How can we create nested frames?
5. Explain different forms of selectors?
6. What is List? Explain different types of lists used in HTML?
7. What is CSS? Explain different mechanisms used to apply CSS to HTML pages?

Unit-II

1. What is JavaScript ? write the features of JavaScript?
2. Write the differences between HTML and DHTML
3. Explain any three objects of JavaScript
4. What is form validation? Explain with example?
5. What is an event ? how can we handle events in JavaScript?
6. What is exception? Explain different ways of handling exceptions in JavaScript?

Unit-III

1. What is XML? Write the differences between XML and HTML
2. What is DTD? Explain the building blocks of DTD?
3. What is XML Schema? Explain the advantages of Schema over DTD?
4. Explain SAX parser?
5. Explain DOM parser? What are the difference between SAX and DOM

Unit_IV

1. What is javaBean ? write the advantages of Java Beans?
2. What is introspection? Explain in detail?
3. What is BDK? Explain the steps involved in BDK tool?
4. Explain persistence and customizers?
5. Explain about EJB Architecture.

Unit-V

1. What is servlet? Explain life cycle of a servlet?
2. What are the advantages of servlets over CGI
3. What is session tracking? Explain different mechanisms of session tracking?
4. Explain classes and interfaces of javax.servlet package
5. Explain classes and interfaces of javax.servlet.http package
6. What are the security issues in servlets?

7. What's the difference between servlets and applets?
8. What is the difference between Difference between doGet() and doPost()?

Unit-VI

1. Write the advantages of JSP over servlets
2. Explain Anatomy of JSP Page
3. What is JSP processing explain with neat diagram?
4. What is MVC explain with neat diagram?
5. What are the difference between JSP and ASP?

Unit-VII

1. Explain different JSP elements?
2. Explain conditional processing with examples?
3. How can we share data between JSP pages?
4. How can we implement session tracking in JSP pages?
5. What are the memory usage considerations

Unit-VIII

1. What is JDBC? Explain the JDBC architecture.
2. Write a JDBC program how to insert and updates records into database.
3. Explain different types of JDBC drivers with suitable examples.
4. Describe the JDBC packages.
5. Compare between JDBC servlets and JDBC JSP with example.

19. Objective Questions:

JNTUK ONLINE EXAMINATIONS [WT]

1. HTML stands for _____.
(a) Hardware Minimized Language
(b) Hardware Markup Language
(c) Hypertext Market Language
(d) Hypertext Markup Language

2. In HTML documents are structured as_____
(a) tags of text
(b) blocks of text
(c) group of text
(d) packets of text

3. The following is an comment tag in HTML documentation
(a) < / this is a comment >
(b) < ! this is a comment >
(c) < this is a comment >
(d) < this is a comment ! >

4. In HTML coding, newlines are treated as_____.
(a) separate line
(b) new page
(c) same line
(d) space

5. In HTML coding, tabs are treated as _____.
(a) same line
(b) space
(c) separate line
(d) new page

6. In HTML white spaces are replaced by_____.
(a) multiple spaces
(b) semicolon
(c) single space
(d) comma

7. The HTML tag used to include ready - formatted text on a web page is _____.
(a) < br >
(b) < amp >
(c) < gre >
(d) < pre >

8. The language used for describing the format of documents which allow them to be viewed on computer screens is _____.
(a) C language (b) COBOL **(c) HTML** (d) Flash

9. HTML formatting is specified by using _____.
(a) Paragraphs (b) Blocks (c) Directives **(d) Tags**

10. The HTML document head holds
(a) control information to be used by the clients
(b) control information to be used by the browsers and servers
(c) data information to be used by the browsers and servers
(d) data information to be used by the clients

11. When the border attribute of a table is set but a valid value is not given, wide default border is drawn.
(a) single centimeter
(b) single pixel
(c) single inch
(d) zero

12. In the following HTML tag < body bgcolor=#nnnnnn text=#nnnnnn link=#nnnnnn vlink=#nnnnnn alink=#nnnnnn > the alink attribute is used to
(a) set the color of a currently inactive link
(b) set the color of a currently active link
(c) set the color of links to be visited
(d) set the color of links visited recently

13. What is the HTML tag used to describe the contents of the table?
(a) < caption > .< /caption >
(b) < table > .< /table >
(c) < desc> .< /desc >
(d) < tabdes > .< /tabdes >

14. The HTML tag used for creating an ordered list is
(a) < ol > .. < /ol >
(b) < ul > .. < /ul >
(c) < dl > .. < /dl>
(d) < li > .. < /li >

15. The type of list which easily incorporates images and paragraphs of text while keeping an obvious structure is _____.
(a) numbered list
(b) definition list
(c) basic bulleted list
(d) character list

16. The HTML tag < img src = "URL1" usemap = "URL2" > tells the browser to _____.
(a) display the image of URL1
(b) display the image of URL2
(c) display the image of URL1 and map the URL2
(d) display the image of URL2 and map the URL1

17. The HTML tag used for creating an unordered list is
(a) < li > .. < /li >
(b) <dl > .. < /dl >
(c) < ol> .. < /ol >
(d) < ul > .. < /ul >

18. What is the component of a web browser which arranges the pieces before the web page is displayed?
(a) search engine
(b) format engine
(c) structure engine
(d) layout engine

19. In the following HTML tag < body bgcolor=#nnnnnn text=#nnnnnn link=#nnnnnn vlink=#nnnnnn alink=#nnnnnn> the vlink attribute is used to
(a) set the color of links to be visited
(b) set the color of a currently active link
(c) set the color of links visited recently
(d) set the color of a currently inactive link

20. The HTML tag used for creating a definition list is
(a) < dl > ..< /dl >
(b) < dd > .. < /dd >
(c) < ul > ..< /ul >
(d) < dt > .. < /dt>

21. In object tag the classid parameter is used to_____.
(a) identify the id
(b) identify the URL of object
(c) identify the object
(d) identify the class

22. A frame based page is actually made from a _____.
(a) set of documents
(b) set of planes
(c) set of websites
(d) set of forms
23. The attributes of frameset tag is
(a) pixel **(b) cols, rows** (c) bgcolor (d) align
24. _____attribute of a frame tag works like an image source or a hyperlink address.
(a) href (b) frame (c) scrolling **(d) src**
25. In displaying the frames the screen can be split _____.
(a) horizontally
(b) diagonally
(c) cannot be split
(d) parallel
26. How many frames can be created in a webpage?
(a) zero
(b) no limit
(c) one
(d) two
27. To ensure that pages are displayed in the correct frame, _____tag is used.
(a) frame
(b) frameset
(c) href
(d) title
28. To display more than one page at a time we use _____.
(a) frameset
(b) blockset
(c) webset
(d) pageset
29. What is the tag used to embed the multimedia objects directly into the webpage?
(a) href
(b) include
(c) link
(d) object
30. If you don't have to have borders on every frame then set the frameborder attribute to_____.
(a) 3
(b) 0
(c) 1
(d) 2
31. In forms to transfer information to a server we use _____.
(a) get
(b) paste
(c) post
(d) put
32. In forms to retrieve information from a server we use _____.
(a) get
(b) put
(c) post
(d) paste
33. In forms data can be sent using .
(a) list
(b) get
(c) assign
(d) put
34. To simply the navigation on complex websites _____is prefered.
(a) frames
(b) layouts
(c) planes
(d) forms
35. In forms, the choice of approach of retrieving the information from a server or transferring information to a server is made by _____attribute.
(a) procedure
(b) function
(c) method
(d) subroutine
36. Data can be sent to forms using .
(a) list (b) put **(c) post** (d) align
37. The attribute of the form tag which specifies the name and location of a CGI script is_____.
(a) method
(b) submit
(c) action
(d) src
38. are used to add an element of interactivity to a website.
(a) planes
(b) frames
(c) forms
(d) layouts
39. To use forms, the server should allow to run .
(a) Java Scripts
(b) XML Scripts
(c) GUI Scripts
(d) CGI Scripts
40. _____are usually used to let reader send information back to the server.
(a) forms
(b) frames
(c) planes
(d) layouts
41. Relative unit em is used for _____.
(a) width of the font for this element
(b) width of the letter
(c) height of the letter
(d) height of the font for this element
42. In cascading stylesheets, to change the appearance of block elements,_____pair of tags are wrapped around the element.
(a) < add > < /add >
(b) < mul > < /mul >
(c) < div > < /div >
(d) < change> </change >
43. Relative unit that is used for height of the font for this element is _____.
(a) ex
(b) en
(c) em
(d) ey
44. The parts of a style rule are _____.
(a) selector, statements
(b) tags, declarations
(c) selector, declarations
(d) scheduler, declarations

45. In cascading stylesheets, to align the text to left we use
(a) text - align: center
(b) text - align: right
(c) text - align: left
(d) text - align: justify
46. In cascading stylesheets, to apply a piece of formatting to many different elements within a page but not necessarily to the entire page, we use_____.
(a) wrapper classes
(b) concrete classes
(c) abstract classes
(d) anonymous classes
47. In a style rule, the selector is used to _____.
(a) create a link between the rules used
(b) create a link between the rule and the HTML tag
(c) create a link between the HTML tags
(d) create a link between the HTML pages
48. In cascading stylesheets the styles are defined in _____.
(a) basic HTML tag
(b) < head > section
(c) internal files
(d) external files
49. A style rule has parts.
(a) two
(b) three
(c) one **Copyright © by UandiStar.org**
(d) four
50. In cascading stylesheets, declarations must be separated using _____and terminated using _____.
(a) colons, semicolons
(b) comma, semicolons
(c) semicolons, comma
(d) semicolons, colons
51. The function that returns the character which is at position index in the string is _.
(a) charOff(index)
(b) charAt(index)
(c) charAt( )
(d) char(index)
52._____is a combination of content formatted using HTML, cascading stylesheets, a scripting language and the DOM.
(a) static HTML
(b) dynamic HTML
(c) HTML
(d) extended HTML
53. In JavaScript, for nesting the strings "JNTU" and "OSMANIA" one inside another and put in University, the correct statement is
(a) university = "JNTU""OSMANIA"
(b) university = "JNTU OSMANIA "
(c) university = "JNTU" + "OSMANIA"
(d) university = JNTU OSMANIA
54. In JavaScript, variables are declared using the keyword_____.
(a) var
(b) variant
(c) varchar
(d) variable
55. JavaScript has only four types of data. They are_____.
(a) integer, float, character, Boolean
(b) numeric, string, Boolean, null
(c) integer, float, string, Boolean
(d) numeric, character, string, Boolean
56. In JavaScript, each line of code is terminated by a _____.
(a) comma
(b) dot
(c) semicolon
(d) colon
57. The scripting language which is platform dependent_____.
(a) VBScript
(b) PerlScript
(c) Jscript
(d) JavaScript
58. JavaScript is developed by_____.
(a) Sun **Copyright © by UandiStar.org (b) Netscape**
(c) Mozilla
(d) Microsoft
59. JavaScript has only four types of data. They are .
(a) numeric, character, string, Boolean
(b) integer, float, string, Boolean
(c) numeric, string, Boolean, null
(d) integer, float, character, Boolean
60. JScript is developed by _____.
(a) Mozilla
(b) Netscape
(c) Microsoft
(d) Sun
61. The variable names in JavaScript can begin with
(a) dollar($)
(b) underscore(_)
(c) minus(-)
(d) carot(^)
62. If the length of the array is n, then the index numbers run from _____.
(a) 0 to n
(b) 1 to n - 1
(c) 0 to n - 1
(d) 1 to n
63. The function floor(5) returns
(a) 4
(b) 0
(c) 5
(d) 6
64. consider the script var first = " A String is" var second = "added to the end" first += second; The contents of the variable first is
(a) String is added to the end
(b) A string is
(c) added to the end A String is
(d) added to the end
65. The function log(-10) base 10 returns
(a) NaN
(b) -1
(c) void
(d) 0

66. JavaScript has _____number of types of operators.
(a) 4
(b) 2
(c) 3
(d) 1
67. String versions of mathematical expressions can be executed in .
(a) floor( )
(b) log( )
(c) eval( )
(d) NaN( )
68. The function atan2(value 1, value 2) returns the arc tangent in .
(a) radians
(b) Fahrenheit
(c) cms
(d) degrees
69. Mathematical functions and values are part of a builtin JavaScript object called____.
(a) math
(b) lang
(c) package
(d) util
70. The JavaScrip numerical constant Math.E is approximately
(a) 2.718
(b) 3.718
(c) 3.148
(d) 2.148
71. In the function parseFloat(string) if the first character of the string does not belong to the valid set, _____isreturned.
(a) void
(b) NaN
(c) 0
(d) 1.0
72. The function floor(5) returns
(a) 6
(b) 5
(c) 4
(d) 0
73. In JavaScript, to inserts a list of elements onto the front of the array, the function is _____.
(a) slice( )
(b) push( )
(c) shift( )
(d) unshift( )
74. In JavaScript the function used to join to arrays is _____.
(a) join( )
(b) shift( )
(c) concat( )
(d) pop( )
75. In JavaScript, to extract a range of elements from an array the function is_____.
(a) shift( )
(b) slice( )
(c) push( )
(d) reverse( )
76. The syntax for creating an array of length 4 in JavaScript is_____.
(a) var data;
(b) var data = [5,6,7];
(c) var data = ["fdgfg", "dfdfd", 3,"345"];
(d) var data = new Array( );
77. In JavaScript, to remove the first element of the array, the function is_____
(a) swap( )
(b) reverse( )
(c) push( )
(d) shift( )
78. In JavaScript, the function which removes the last element from the array is _____.
(a) concat( )
(b) pop( )
(c) shift( )
(d) join( )
79. In JavaScript, to alter an array by removing some elements the function is .
(a) splice()
(b) slice( )
(c) reverse( )
(d) sort( )
80. In JavaScript the function used to join all the elements of an array as a string is_____.
(a) join( )
(b) pop( )
(c) shift( )
(d) concat( )
81. In JavaScript array is a _____.
(a) object
(b) stack
(c) class
(d) function
82. In JavaScript the function which swaps all of the elements in the array so that which was first is last and vice versa.
(a) push( )
(b) swap( )
(c) reverse( )
(d) shift( )
83. The javascript is loaded by the browser but isnt run until the_____action of the button is triggered.
(a) press( )
(b) onclick( )
(c) onpress( )
(d) click( )
84. _____may be used as a warning or to provide a farewell message as visitors leave your site.
(a) prompt("string", "string")
(b) alert("string")
(c) prompt("string")
(d) confirm("string")
85. forms have an _____event which is generated when the submit button is clicked.
(a) insubmit
(b) oversubmit
(c) butsubmit
(d) onsubmit
86. _____shows a window containing a message and two buttons: Ok and Cancel.
(a) prompt("string", "string")
(b) confirm("string")d.
(c) alert("string")
(d) rompt("string")b.

87. _____frame is said to be the parent of all of the internal frames.
(a) application
(b) parent
(c) child
(d) presentation
88. _____is the process of ensuring that some data might be correct data for a particular application.
(a) Data validation
(b) Data correction
(c) Data checking
(d) Data deletion
89. _____command displays a simple window that contains a prompt and a textfield in which the user can enter data.
(a) confirm("string")
(b) prompt("string", "string")
(c) alert("string")
(d) prompt("string")
90. If the script needs to act upon the browser window in which it is running, then is used.
(a) itself
(b) ourselves
(c) self
(d) themselves
91. _____is the process of ensuring that users submit only the set of characters which you require.
(a) Data correction
(b) Data checking
(c) Data validation
(d) Data deletion
92. _____displays the helpful information about the operation of the browser.
(a) status bar
(b) display bar
(c) task bar
(d) view bar
93. When positioning the logo in Netscape, the left edge will be_____in from the right side of the screen.
(a) 360 pixels
(b) 240 pixels
(c) 120 pixels
(d) 720 pixels
94. Javascript rollover uses number of image files which it swaps between as the mouse is moved.
(a) 3 **(b) 2** (c) 4 (d) 1
95. _____ tells the browser what to do when certain events happen.
(a) StartupEvents( )
(b) SetupEvents( )
(c) BeginEvents( )
(d) CloseEvents( )
96. Rollover buttons, Moving Images, etc 1. The technique used to give visual feedback about the location of the mouse cursor by changing the images on the page as the mouse moves over them is _____.
(a) image passover
(b) image moving
(c) image crossover
(d) image rollover
97. When positioning the logo in Netscape the available width is the _____.
(a) width of the window the vertical scroll
(b) width of the window + the vertical scroll
(c) width of the window the horizontal scroll
(d) width of the window + the horizontal scroll
98._____calls a function when the cursor moves away from the image.
(a) onmouserover
(b) onload
(c) onmouseout
(d) onsubmit
99. The_____happens when the page is first loaded into the browser.
(a) onload
(b) onmouserover
(c) onsubmit
(d) onmouseout
100. When positioning the logo in Netscape, the top of the logo will appear at .
(a) 50 pixels above the top of the screen
(b) 100 pixels below the bottom of the screen
(c) 100 pixels above the top of the screen
(d) 100 pixels below the bottom of the screen
101. When positioning the logo in Netscape the available height is the_____- .
(a) height of the window + the vertical offset of the page
(b) height of the window + the horizontal offset of the page
(c) height of the window the vertical offset of the page
(d) height of the window the horizontal offset of the page
102. _____calls a javascript function when the cursor passes the over the image.
(a) onsubmit
(b) onmouserover
(c) onload
(d) onmouseout
103. In XML document, the data type of the element must be preceded by a ____symbol.
(a) # (b) & (c) $ (d) ^
104._____is used to describe the structure of a document not the way that it is presented.
(a) PDF
(b) XML
(c) HTML
(d) RTF
105. For creating own markup languages which conform to an international standards and can be manipulated by many applications but which are exactly tailored to a specific set of needs is_____.
(a) DHTML
(b) ZML
(c) XML
(d) HTML
106. XML elements basically hold one of two data types
(a) PDATA, CDATA
(b) CPDATA, CDATA
(c) PCDATA, CDATA
(d) ADATA, CDATA
107. The scripting language which is used to "describe how the data looks and also gives

information about what it is" is _____.
(a) HTML (b) PDF (c) RTF **(d) XML**
108. IN XML, the element tag starts with _____.
(a) caret
(b) asterisk mark
(c) ampersand symbol
(d) exclamation mark
109. The first node of the XML document is called _____.
(a) start node
(b) node
(c) root node
(d) leaf node
110. XML documents are composed of the following things.
(a) elements, flow information, entities
(b) elements, control information, entities
(c) objects, flow information, elements
(d) objects, control information, entities
111. _____are used to control applications in XML documents.
(a) processing instructions
(b) debugging instructions
(c) control instructions
(d) execution instructions
112._____is a sort of meta-markup; a grammar for creating other markup languages.
(a) HTML
(b) XML
(c) DHTML
(d) RTF
113. _____model is used when passing XML data across a network between applications and is widely used by java programmers.
(a) XAS **(b) SAX** (c) MOD (d) DOM
114. In XML,_____are used to create small pieces of data which you want to use repeatedly throughout your schema.
(a) backward entities
(b) external entities
(c) internal entities
(d) forward entities
115. All complex data items which do not need to pass through the XML parser should be defined as
(a) non-parsed
(b) parsed
(c) simple
(d) complex
116. The DOM views the XML documents are_____ .
(a) layers
(b) pointers
(c) trees
(d) pages
117. Almost anything which is data can be included in your XML as an_____.
(a) internal entity
(b) backward entity
(c) external entity
(d) forward entity
118. _____is an application program interface(API) for XML documents.
(a) XAS model
(b) SAX model
(c) MOD model
(d) DOM model
119. SAX parsers are used when dealing with _____.
(a) characters of data
(b) bytes of data
(c) streams of data
(d) bits of data
120. On web sites where repeated querying and updating of the XML document is required, we use _____.
(a) SAX model
(b) XAS model
(c) MOD model
(d) DOM model
121. The_____exposes the whole of the document to applications.
(a) SAZ (b) MOD **(c) DOM** (d) SAX
122. A is a way of keeping the names used by applications separate from each other.
(a) namesdata
(b) nameloop (c)
namebuffer **(d)**
namespace
123. SAX parsers are used when dealing with _.
(a) bytes of data
(b) characters of data
(c) bits of data
(d) streams of data
124. To define a single template for output based on a specific pattern, the command is
(a) xsl : eval
(b) xsl : attribute
(c) xsl : template
(d) xsl : scrit
125. The following command is used to create an attribute node.
(a) xsl : choose
(b) xsl : cdata
(c) xsl : apply - templates
(d) xsl : attribute
126. The following command is used to add CDATA section to the output document.
(a) xsl : choose
(b) xsl : cdata
(c) xsl : apply templates
(d) xsl : attribute
127. To transform one data structure into another _____is used.
(a) XML
(b) DHTML
(c) HTML
(d) XSL
128. The _____is a language used to express stylesheets which are then used to present XML documents.
(a) Intensible Stylesheet
(b) Formatting Stylesheet
(c) Cascading Stylesheet

(d) Extensible Stylesheet

129. The following command directs the processor towards the most appropriate template for the situation

(a) xsl : apply templates
(b) xsl : choose
(c) xsl : attribute
(d) xsl : cdata

130. To check the condition of an element, the following command is used.

(a) xsl : apply templates
(b) xsl : cdata
(c) xsl : attribute
(d) xsl : choose

131. To declare global variables and functions within a template, the command is

(a) xsl : template
(b) xsl : attribute
(c) xsl : script
(d) xsl : cdata

132. Using the following command, a single template is applied to a set of XML elements.

(a) xsl : choose
(b) xsl : apply templates
(c) xsl : attribute
(d) xsl : for – each

133. To copy the target node from the input source to the output, the following command is used.

(a) xsl : comment
(b) xsl : apply - templates
(c) xsl : choose
(d) xsl : cdata

134. To transform XML document into HTML_____is required.

(a) Perl (b) HTML (c) XML (d) DHTML

135. In XML::Parser, _____handler is called for all characters which do not have a specified handler either because they are not part of the markup or because no handler has been registered for them.

(a) Proc(Expat, target, data)
(b) Default(Expat, string)
(c) Comment(Expat, data)
(d) ExternEnt(Expat, base, sysid, pubid)

136. In XML::Parser, _handler is used when a processing instruction(PI) has been found.

(a) ExternEnt(Expat, base, sysid, pubid)
(b) Proc(Expat, target, data)
(c) Comment(Expat, data)
(d) Default(Expat, string)

137. In parsing XML, XML cannot be handled with regular expressions because

(a) XML elements will often span a number of lines of text.
(b) Perl regular expressions can handle arbitrary nesting
(c) White spaces and newlines have meaning in XML
(d) Matching pairs of tags is straightforward in complex documents

138. In transforming XML into HTML, querying and updating capabilities are there in ___.

(a) XML (b) XSL **(c) Perl** (d) Corba

139. In transforming XML into HTML, if we use Perl for these tasks then _____.

(a) server and client share equally the work
(b) most of the work is done by the client
(c) most of the work is done by the server
(d) there is no task to be performed by the server

140. In XML::DOM, the constant value '3' represents

(a) ATTRIBUTE_NODE
(b) ELEMENT_NODE
(c) TEXT_NODE
(d) UNKNOWN_NODE

141. The XML::DOM module is a compliant parser.

(a) DOM level III
(b) DOM level I
(c) DOM level IV
(d) DOM level II

142. The DOM parser creates a _____data structure composed of nodes.

(a) stack
(b) heap
(c) tree
(d) linked list

143. In XML::DOM::Element class, _method is used to return the value of a named attribute.

(a) getTagName( )
(b) setTagName(name)
(c) getAttribute(name)
(d) setAttribute(name, value)

144. In XML::DOM::Node class, the method which returns an integer indicating the type of the current node is

(a) getNodeNumber
(b) getNodeType
(c) getNodeFormat
(d) getNodeName

145. In XML::DOM, the constant value '1' represents

(a) ELEMENT_NODE
(b) ATTRIBUTE_NODE
(c) TEXT_NODE
(d) UNKNOWN_NODE

146. In XML::DOM, the value of DOCUMENT_FRAGMENT_NODE is

(a) 12 (b) 9 (c) 10 **(d) 11**

147. In XML::DOM::Element class, _____ method is used to return the name of the element.

(a) getAttribute(name)
(b) setTagName(name)
(c) getTagName( )
(d) setAttribute(name, value)

148. _____class is a collection of nodes which can be accessed directly via their name.

(a) XML::DOM::Node
(b) XML::DOM::NamedNodeMap
(c) XML::DOM::Element
(d) XML::DOM::Text

149. In XML::DOM, the value of CDATA_SECTION_NODE is

(a) 4 (b) 6 (c) 7 (d) 5

150. A Java Bean may be _____to an end user.

(a) visible
(b) invisible
(c) transparent
(d) visible or invisible

151. _____provides the component based architecture.
(a) Java Packages
(b) Java Applets
(c) Java Servlets
(d) Java Beans

152._____turns classes into software components by providing several new features.
(a) Java Packages
(b) Java Beans
(c) Java Servlets
(d) Java Applets

153. A Bean is a Java Beans_____.
(a) interface (b) classb **(c) component** (d) object

154. The configuration settings of a Java Bean can be saved in _____storage.
(a) Permanent
(b) Temporary
(c) Persistent
(d) Secondary

155. _____independent, reusable software modules.
(a) Java Applets
(b) Java Servlets
(c) Java Classes
(d) Java Beans

156. _____is a software component that has been designed to be reusable in a variety of different environments.
(a) Java Packages
(b) Java Servlets
(c) Java Beans
(d) Java Applets

157. A Java Bean obtains all the benefits of Java's_____paradigm.
(a) write - twice, run - twice
(b) write - twice, run - anywhere
(c) write - anywhere, run - anywhere
(d) write - once, run – anywhere

158. Java Beans architecture consists of the following things.
(a) objects, functions, datatypes
(b) objects, methods, properties
(c) objects, methods, properties
(d) events, properties, methods

159. _____ tool enables you to create, configure and connect a set of Beans.
(a) BDK (b) AWT (c) JDK (d) SDK

160. The utility to generate a jar file is _____.
(a) Files jar (b) Jar **(c) jar options files** (d) files

161. The command to list the contents of Xyz.jar file is
(a) jar pf Xyz.jar
(b) jar mf Xyz.jar
(c) jar tf Xyz.jar
(d) jar cf Xyz.jar

162. A_____directory contains Java Beans and Java Beans Related class and interface documentation is
(a) apis (b) readme **(c) javadoc** (d) demo

163. The command to extract the contents of Xyz.jar and place those files in the current directory is
(a) jar mf Xyz.jar
(b) jar pf Xyz.jar
(c) jar xf Xyz.jar
(d) jar tf Xyz.jar

164. The command that adds the file file1.class to Xyz.jar
(a) jar file1.class Xyz.jar
(b) jar cf Xyz.jar file1.class
(c) jar uf file1.class Xyz.jar
(d) jar uf Xyz.jar file1.class

165. _____file is used to indicate which of the components in a JAR file are java beans.
(a) multimedia
(b) manifest
(c) jar
(d) doc

166. The command to add all files below directoryX to Xyz.jar
(a) jar - mf Xyz.jar - u directoryX.jar
(b) jar - cf Xyz.jar - c directoryX.jar
(c) jar - cf Xyz.jar - u directoryX.jar
(d) jar - uf Xyz.jar - c directoryX.jar

167. The command to create a JAR file name Xyz.jar that contains all of the .class and .gif files in the current directory is
(a) jar bf Xyz.jar *.class *.gif
(b) jar cf Xyz.jar *.class *.gif
(c) jar mf Xyz.jar *.class *.gif
(d) jar af Xyz.jar *.class *.gif

168. The_____directory contains makefie fpr building the demo Beans.
(a) apis
(b) Javadoc
(c) Beans/Demos
(d) readme

169. The method in which the developer of a Bean can indicate which of its properties, events, and methods should be exposed by an application builder tool is _____.
(a) no name convention
(b) extends the BeanInfo class
(c) importing a datatype
(d) extends the BeanInfo interface

170. The process of analyzing a Bean to determine its capabilities is
(a) Introspection (b) Interruption
(c) Intersection (d) Interception

171. In Java Beans, the design pattern for Indexed property is identified by_____, where N is the name of the property,T is its type.
(a) public T getN(int index);
public void setN(int index, T value);
public T [ ] getN( );
public void setN(T values[ ]);
(b) public N get(int index);
public void setN(int index, T value);
public T [ ] getN( );
public void setN(T values[ ]);
(c) public T getT(int index);
public void setT(int index, N value);
public N[ ] getN( );

public void setN(T values[ ]);
(d) public N getN(int index);
public void setT(index, T value);
public T[] get( );
public void setN(T values[ ]);

172. The introspection mechanism finds all of the_____methods of a bean.
(a) public (b) protected (c) owned (d) private

173. In Java Beans, the design pattern for Simple property is identified by_____, where N is the name of the property,T is its type.
(a) public T getT( ); public void setN(N arg);
(b) public N getN( ); public void setT(T arg);
(c) public T getN( ); public void setN(T arg);
(d) public N getT( ); public void setT(N arg);

174. The properties of a Bean are set through_____method.
(a) installer
(b) getter
(c) implementer
(d) setter

175. _____allows interrogation of a Bean classes properties and methods.
(a) Introspection
(b) Interruption
(c) Interception
(d) Intersection

176. To obtain information about a component the design tool used is
(a) Introspection
(b) Interruption
(c) Intersection
(d) Interception

177. The properties of a bean are obtained by_____method.
(a) implementer **(b) getter** (c) installer (d) setter

178. _____interface allows a designer to provide a graphical user interface through which a Bean may be configured.
(a) Customizer
(b) BeanInfo
(c) Visibility
(d) DesignMode

179. _____class encapsulates a call to a method that returns a result.
(a) Encoder **(b) Expression**
(c) Beans (d) FeatureDescriptor

180. class analyzes a Bean and constructs a BeanInfo object that describes the component.
(a) Encoder (b) Expression
(c) Beans **(d) Introspector**

181. _____class is used to obtain information about a Bean.
(a) InfoBean
(b) Beans
(c) BeanEncoder
(d) BeanInfo

182. When creating a class that implements BeanInfo, you must call that class_____ where bname is the name of the Bean.
(a) bnameBeanInfo
(b) BeanInfobname
(c) Beanbname
(d) bnameBean

183. To simplify the use of BeanInfo, JavaBeans supplies the _____class.
(a) BeanInfoSimple
(b) BeanInfo
(c) SimpleBean
(d) SimpleBeanInfo

184._____enables you to implicitly determine what information is available to the user of a Bean.
(a) Design arrays
(b) BeanInfo class
(c) BeanInfo interface
(d) Design patterns

185. _____interface allows a designer to specify information about the properties, events, and methods of a Bean.
(a) DesignMode
(b) Visibility
(c) BeanInfo
(d) Customizer

186. _____enables you to explicitly control what information is available.
(a) BeanInfo Interface
(b) BeanInfo package
(c) BeanInfo class
(d) BeanInfo Applet

187. The classes PropertyDescriptor, EventSetDescriptor and MethodDescriptor are defined within the _____package.
(a) java .io.beans
(b) java.awt.beans
(c) java.beans
(d) java.applet.beans

188. A Bean that has a_____property generates an event when an attempt is made to change its value.
(a) fixed
(b) restricted
(c) bound
(d) constrained

189. A _____can provide step - by - step guide through the process that must be followed to use the component in a specific context.
(a) assistance
(b) persister
(c) helper
(d) customizer

190. A class that handles PropertyChangeEvent event must implement the____interface.
(a) PropertyChangeListener
(b) PropertyChangeBean
(c) PropertyChangeInterface
(d) PropertyChangeEvent

191. A bean that has a_____property generates an event when the property is changed.
(a) bound (b) fixed (c) restricted (d) constraine

192. The easiest way to serialize a Bean is to have it implement the_____interface.
(a) java.io.
(b) java.io.Serializable
(c) java.Serializable
(d) java.Serializable.io

193. A bean developer can provide a _____ that helps another developer configure the Bean.
(a) customizer
(b) persister
(c) helper
(d) assistance
194. _____is the ability to save the current state of a Bean.
(a) saving
(b) persistence
(c) storing
(d) permanent
195. A class that handles the PropertyVetoException event must implement the_____ interface.
(a) ChangeVetoListener
(b) VetoableChangeListener
(c) VetoChangeableListener
(d) NovetoListener
196. The objects have the ability to veto the proposed change by throwing a_____.
(a) VetoPropertyException
(b) VetoException
(c) PropertyVetoException
(d) VetoProperty
197. A Java Bean is a software component that has been designed to be_____.
(a) reusable
(b) reversible
(c) redesign
(d) reengineering

20. Tutorial problems

1. Discuss about the web technologies software applications.
2. What is CSS? How many types of CSS and Explain.
3. What are difference between HMTL and JavaScript? Explain with example.
4. What are the form validations? Explain with example.
5. What is XML? What are difference between XML and HMTL.
6. Define the following terms
   i). DHTML ii). DTD iii). SAX iv). DOM
7. What is javaBean ? Write the advantages of Java Beans?
8. Explain about EJB architecture and process.
9. What is servlet? Explain life cycle of a servlet?
10. What is session tracking? Explain different mechanisms of session tracking?
11. Explain classes and interfaces of javax.servlet.http package
12. What is the difference between Difference between doGet() and doPost()?

21. Known Gaps

No

22.Discussion topics

1. Insertion of images into Frames
2. Regular expressions in JavaScript
3. Naming space in XML
4. BDK Introspection
5. Enterprise Java Beans
6. Tomcat Installation
7. Servlet API
8. Java Server Page applications with MVC architecture
9. Error handling and Debugging in JSP
10. JDBC drivers
11. Javax.sql.* package

23. References, Journals, websites and E-links

TEXT BOOKS

1. Web Programming, building internet applications, Chris Bates 2nd edition, WILEY Dreamtech (UNIT s 1,2 ,3)
2. The complete Reference Java 2 Fifth Edition by Patrick Naughton and Herbert Schildt. TMH (Chapters: 25) (UNIT 4)
3. Java Server Pages –Hans Bergsten, SPD O'Reilly (UNITs 5,6,7,8)

REFERENCES

1. Programming world wide web-Sebesta,Pearson
2. Core SERVLETS ANDJAVASERVER PAGES VOLUME 1: CORE TECHNOLOGIES By Marty Hall and Larry Brown Pearson
3. Internet and World Wide Web – How to program by Dietel and Nieto PHI/Pearson Education Asia.
4. Jakarta Struts Cookbook , Bill Siggelkow, S P D O'Reilly for chap 8.
5. Murach's beginning JAVA JDK 5, Murach, SPD
6. An Introduction to web Design and Programming –Wang-Thomson
7. Web Applications Technologies Concepts-Knuckles,John Wiley
8. Programming world wide web-Sebesta,Pearson

WEBSITES

1. www.Wikipedia/web technologies.com
2. www. w3cschool.com
3. www.sun.java.com
4. www.javatpoint.com

JOURNALS

1. Emerging Technologies in Web Intelligence
2. International Journal of Web Engineering and Technology
   Using Provenance in the Semantic Web

24.Course Assessment

Geethanjali College of Engineering and Technology

CHEERYAL (V), KEESARA (M), R.R.DIST-501301, ANDHRAPRADESH

Department of Computer Science & Engineering

Course Assessment

Class: III year A.Y: 2014-15

Subject: WT Sem: II

Faculty: S.RAMANJANEYULU SECTION: C

Assessment	Criteria Used	Attainment Level			Remarks
Direct(d)	**Theory:**				
	External Marks	92.59	92.32	95.69	
	Internal Marks(Theory)	92.59			
	Assignments	100			
	Tutorials	84.10			
	Lab:				
	Internal Marks	100	99.07		
	External Marks	98.14			
Indirect(id)	Course End Survey			85.38	
Course Assessment(0.6*d+0.4*id)				91.56	

25.Student List

26. Group-Wise students list for discussion topics