DIGITAL NOTES
ON
(R15A0521)WEB TECHNOLOGIES

B.TECH III YEAR - I SEM
(2018-19)

DEPARTMENT OF INFORMATION TECHNOLOGY

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)
(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, INDIA.

III Year B.Tech. IT - I Sem

L	T/P/D	C
5	1/-/-	4

(R15A0521)WEB TECHNOLOGIES

Objectives:

1. Giving the students the insights of the Internet programming and how to design and implement complete applications over the web.
2. It covers the notions of Web servers and Web Application Servers, Design Methodologies with concentration on Object-Oriented concepts, Client-Side
3. Programming, Server-Side Programming, Active Server Pages, Database Connectivity to web applications, Adding Dynamic content to web applications,
4. Programming Common Gateway Interfaces, Programming the User Interface for the web applications.

UNIT I:

Web Basics and Overview: Introduction to Internet, World wide web, Web Browsers, URL, MIME, HTTP, Web Programmers Tool box.

HTML Common tags: List, Tables, images, forms, Frames; Cascading Style sheets. Introduction to Java Scripts, Objects in Java Script, Dynamic HTML with Java Script.

UNIT II:

Introduction to XML: Document type definition, XML Schemas, Document Object model, Presenting XML ,Introduction to XHTML, Using XML Processors: DOM and SAX.

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 III:

Web Servers and Servlets: Tomcat web server, Installing the Java Software Development Kit, Tomcat Server & Testing Tomcat, Introduction to Servlets: Lifecycle of a Servlet, JSDK, The Servlet API, The javax. Servlet Package, Reading Servlet 150 parameters, Reading Initialization parameters. The javax.servlet HTTP package, Handling Http Request & Responses, Using Cookies-Session Tracking, Security Issues.

UNIT IV:

Database Access: Database Programming using JDBC, JDBC drivers, Studying Javax.sql.* package, Accessing a Database from a Servlet. Introduction to JSP: The Problem with Servlet. The Anatomy of a JSP Page, JSP Processing. JSP Application Design with MVC Setting Up and JSP Environment.

UNIT V:

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, Accessing a Database from a JSP page, 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)
2. Core SERVLETS ANDJAVASERVER PAGES VOLUME 1: CORE TECHNOLOGIES By Marty Hall and Larry Brown Pearson (UNITs 3,4,5)

REFERENCE BOOKS:

1. Programming world wide web-Sebesta,Pearson Education ,2007.
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. March's beginning JAVA JDK 5, Murach, SPD
6. An Introduction to Web Design and Programming –Wang-Thomson

Course Outcomes:

1. Analyze a web page and identify its elements and attributes.
2. Create web pages using XHTML and Cascading Styles sheets.
3. Installation and usage of Server software's.
4. Database Connectivity to web applications
5. Build web applications using Servlet and JSP

INDEX

UNIT-I

Introduction to Internet:- A global computer network providing a variety of information and communication facilities, consisting of interconnected networks using standardized communication protocols. "the guide is also available on the Internet"

The Internet is the global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) to link devices worldwide. It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies. The Internet carries a vast range of information resources and services.

World Wide Web

The World Wide Web (abbreviated WWW or the Web) is an information space where documents and other web resources are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and can be accessed via the Internet. English scientist Tim Berners-Lee invented the World Wide Web in 1989. He wrote the first web browser computer program in 1990 while employed at CERN in Switzerland. The Web browser was released outside CERN in 1991, first to other research institutions starting in January 1991 and to the general public on the Internet in August 1991.

Browsers:

WWW Clients, or "Browser": The program you use to access the WWW is known as a browser because it "browses" the WWW and requests these hypertext documents. Browsers can be graphical, allows to see and hear the graphics and audio

Uniform Resource Locators, or URLs:

A Uniform Resource Locator, or URL is the address of a document found on the WWW. Browser interprets the information in the URL in order to connect to the proper Internet server and to retrieve your desired document. Each time a click on a hyperlink in a WWW document instructs browser to find the URL that's embedded within the hyperlink.
The elements in a URL: **Protocol://server's address/filename**

Hypertext protocol: http://www.aucegypt.edu

File Transfer Protocol: ftp://ftp.dartmouth.edu

Telnet Protocol: telnet://pac.carl.org

News Protocol: news:alt.rock-n-roll.stones

What are Domains? Domains divide World Wide Web sites into categories based on the nature of their owner, and they form part of a site's address, or uniform resource locator (URL).

- Mime (multi-purpose internet mail extensions):-
- Hypertext transport protocol:

HTML Common tags:-

HTML is the building block for web pages. HTML is a format that tells a computer how to display a web page. The documents themselves are plain text files with special "tags" or codes that a web browser uses to interpret and display information on your computer screen.

HTML stands for Hyper Text Markup Language

An HTML file is a text file containing small markup tags

The markup tags tell the Web browser how to display the page

An HTML file must have an htm or html file extension.

Tag	Description
<!DOCTYPE...>	This tag defines the document type and HTML version.
<html>	This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.
<head>	This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.
<title>	The <title> tag is used inside the <head> tag to mention the document title.
<body>	This tag represents the document's body which keeps other HTML tags like <h1>, , <p> etc.
<p>	This tag represents a paragraph.
<h1> to <h6>	Defines header 1 to header 6
 	Inserts a single line break
<hr>	Defines a horizontal rule
<!-->	Defines a comment

CSS stands for Cascading Style Sheets

CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**.
CSS **saves a lot of work**. It can control the layout of multiple web pages all at once.
CSS can be added to HTML elements in 3 ways:

- **Inline** - by using the style attribute in HTML elements
- **Internal** - by using a <style> element in the <head> section
- **External** - by using an external CSS file

Inline CSS
An inline CSS is used to apply a unique style to a single HTML element.

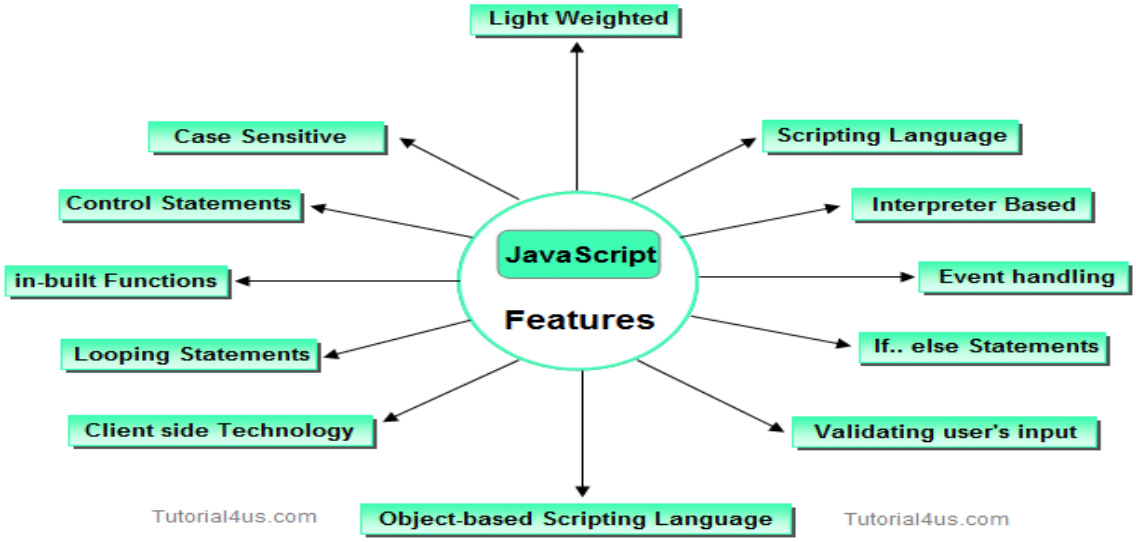
An inline CSS uses the style attribute of an HTML element.

JavaScript:

What is JavaScript?

Java Script is one popular scripting language over internet. Scripting means a small sneak (piece). It is always independent on other languages.

JavaScript is most commonly used as a client side scripting language. This means that JavaScript code is written into an HTML page. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it's up to the browser to do something with it.



UNIT-II

XML - XML stands for **Extensible Mark-up Language**, developed by W3C in 1996. It is a text-based mark-up language derived from Standard Generalized Mark-up Language (SGML). XML 1.0 was officially adopted as a W3C recommendation in 1998. XML was designed to carry data, not to display data. XML is designed to be self-descriptive. XML is a subset of SGML that can define your own tags. A Meta Language and tags describe the content. XML Supports CSS, XSL, DOM. XML does not qualify to be a programming language as it does not performs any computation or algorithms. It is usually stored in a simple text file and is processed by special software that is capable of interpreting XML.

XML Schemas

- ✓ XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database. XSD extension is **".xsd"**.
- ✓ This can be used as an alternative to XML DTD. The XML schema became the W#C recommendation in 2001.
- ✓ XML schema defines elements, attributes, element having child elements, order of child elements. It also defines fixed and default values of elements and attributes.
- ✓ XML schema also allows the developer to us **data types**.

Document Object Model

The Document Object Model protocol converts an XML document into a collection of objects in your program. XML documents have a hierarchy of informational units called nodes; this hierarchy allows a developer to navigate through the tree looking for specific information. Because it is based on a hierarchy of information, the DOM is said to be tree based. DOM is a way of describing those nodes and the relationships between them.

You can then manipulate the object model in any way that makes sense. This mechanism is also known as the "random access" protocol, because you can visit any part of the data at any time. You can then modify the data, remove it, or insert new data.

The XML DOM, on the other hand, also provides an API that allows a developer to add, edit, move, or remove nodes in the tree at any point in order to create an application. A DOM parser creates a tree structure in memory from the input document and then waits for requests from client. A DOM parser always serves the client application with the **entire document no matter how much is actually needed** by the client. With DOM parser, method calls in client application have to be explicit and forms a kind of chained method calls.

Document Object Model is for defining the standard for accessing and manipulating XML documents. **XML DOM** is used for

- **Loading the xml document**
- **Accessing the xml document**
- **Deleting the elements of xml document**
- **Changing the elements of xml document**

XHTML: eXtensible Hypertext Markup Language

Hypertext is simply a piece of text that works as a link. **Markup language** is a language of writing layout information within documents. The XHTML recommended by **W3C**. Basically an XHTML document is a plain text file and it is very much similar to HTML. It contains rich text, means text with tags. The extension to this program should b either **html or htm**. These programs can be opened in some web browsers and the corresponding web page can be viewed.

HTML Vs XHTML

HTML	XHTML
1. **The HTML tags are case insensitive.** **EX: <BoDy>---------</body>**	**1.** The XHTML tags are case sensitive. EX: <body>---------</body>
2. **We can omit the closing tags sometimes.**	**2.** For every tag there must be a closing tag. EX: --------- or <h1----------/>
3. **The attribute values not always necessary to quote.**	**3.** The attribute values are must be quoted.
4. **In HTML there are some implicit attribute values.**	**4.** In XHTML the attribute values must be specified explicitly.
5. **In HTML even if we do not follow the nesting rules strictly it does not cause much difference.**	**5.** In XHTML the nesting rules must be strictly followed. These nesting rules are- - A form element cannot contain another form element. -an anchor element does not contain another form element -List element cannot be nested in the list element -If there are two nested elements then the inner element must be enclosed first before

	closing the outer element -Text element cannot be directly nested in form element

Differences between DOM and SAX

DOM	SAX
Stores the entire XML document into memory before processing	Parses node by node
Occupies more memory	Doesn't store the XML in memory
We can insert or delete nodes	We can't insert or delete a node
DOM is a tree model parser	SAX is an event based parser
Document Object Model (DOM) API	SAX is a Simple API for XML
Preserves comments	Doesn't preserve comments
DOM is slower than SAX, heavy weight.	SAX generally runs a little faster than DOM, light weight.
Traverse in any direction.	Top to bottom traversing is done in this approach
Random Access	Serial Access
Packages required to import import javax.xml.parsers.*; import javax.xml.parsers.DocumentBuilder; import javax.xml.parsers.DocumentBuilderFactory;	**Packages required to import** import java.xml.parsers.*; import org.xml.sax.*; import org.xml.sax.helpers;

JavaBeans

JavaBeans is architecture for both using and building components in Java. This architecture supports the features of software reuse, component models, and object orientation. One of the most important features of JavaBeans is that it does not alter the existing Java language.

Although Beans are intended to work in a visual application development tool, they don't necessarily have a visual representation at run-time (although many will). What this does mean is that Beans must allow their property values to be changed through some type of visual interface, and their methods and events should be exposed so that the development tool can write code capable of manipulating the component when the application is executed.

Bean Development Kit (BDK) is a tool for testing whether your JavaBeans meets the JavaBean specification.

BDK (Bean Development Kit)

The Bean Development kit is a tool that allows the user to configure and interconnect a set of beans. The user can change the properties of a Bean, link two or more Beans and execute Beans.

Start the "beanbox" by running "$bdk\beanbox\run.bat".

The Bean Development Kit (BDK) represents a Toolbox, a Bean Box and a Property Window

Bound – A bean property for which a change to the property results in a notification being sent to some other bean.

Constrained – A bean property for which a change to the property results in validation by another bean. The other bean may reject the change if it is not appropriate.

Bound Properties:

Bound properties are the properties that can be connected to another property of the same type in a

different bean, so that when the source bean's property value changes, the target bean's property changes to the same value. But some beans only have these bound properties. For example put the jelly bean in our composite BeanBox.

UNIT-III

Web servers and servlets: **Web Servers:** Any computer can be turned into a Web server by installing server software and connecting the machine to the Internet. There are many Web server software applications, including public domain software and commercial packages.

Install TOMCAT web server and APACHE.

While installation, we assign port number 8080 to APACHE. Make sure that these ports are

Web servers are computers that deliver (*serves up*) Web pages. Every Web server has an IP address and possibly a domain name. For example, if you enter the URL *http://www.mrcet.com/index.html* in your browser, this sends a request to the Web server whose domain name is mrcet.*com*. The server then fetches the page named *index.html* and sends it to your browser.

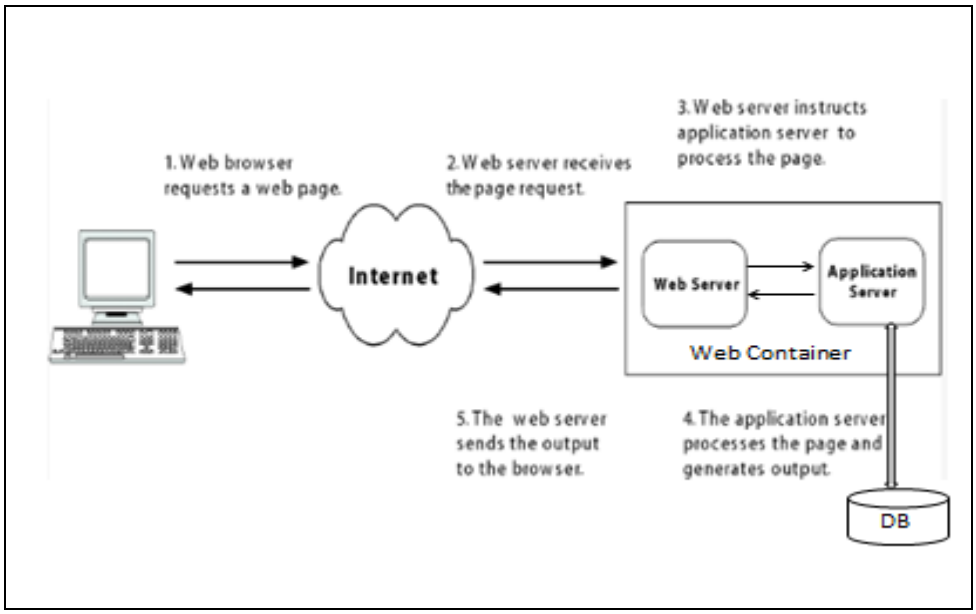
available i.e., no other process is using this port.

INTRODUCTION TO SERVLETS

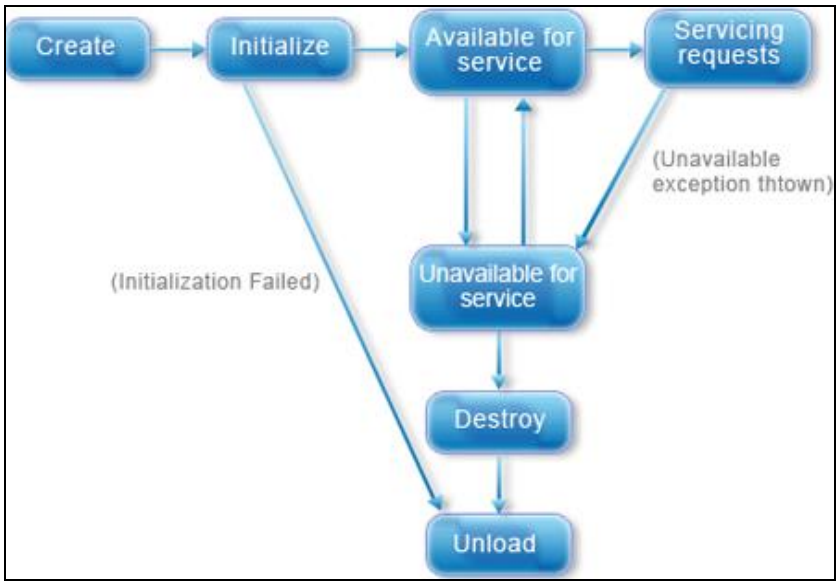
Servlets:

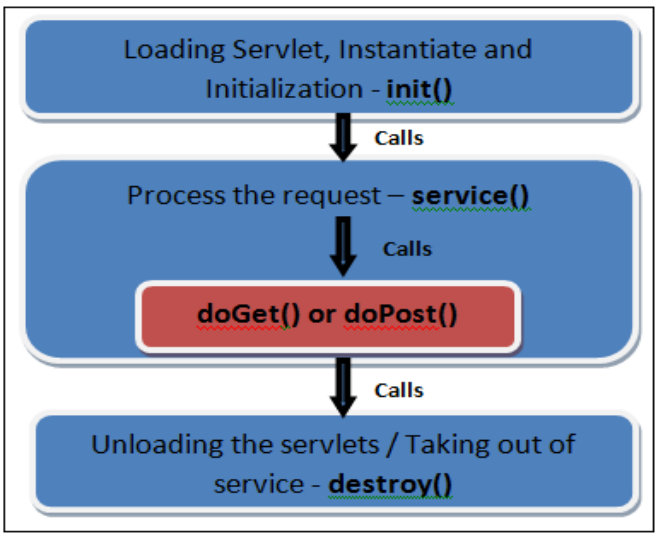
- Servlets are server side programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser and databases or applications on the server.
- Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
- Servlets don't fork new process for each request, instead a new thread is created.
- Servlets are loaded and ready for each request.
- The same servlet can handle many requests simultaneously.

Web Container: It is web server that supports servlet execution. Individual Servlets are registered with a container. Tomcat is a popular servlet and JSP container.



Life Cycle of Servlet





Servlet API

Servlet API consists of two important packages that encapsulates all the important classes and interface, namely :

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

1. javax.servlet

Interfaces

1. Servlet – Declares life cycle methods for a servlet.
2. ServletConfig – To get initialization parameters
3. ServletContext- To log events and access information
4. ServletRequest- To read data from a client request
5. ServletResponse – To write data from client response

Classes

1. GenericServlet – Implements Servlet and ServletConfig
2. ServletInputStream – Provides an input stream for reading client requests.
3. ServletOutputStream - Provides an output stream for writing responses to a client.
4. ServletException – Indicates servlet error occurred.
5. UnavailableException - Indicates servlet is unavailable

Servlet API

Servlet API consists of two important packages that encapsulates all the important classes and interface, namely :

3. **javax.servlet**
4. **javax.servlet.http**

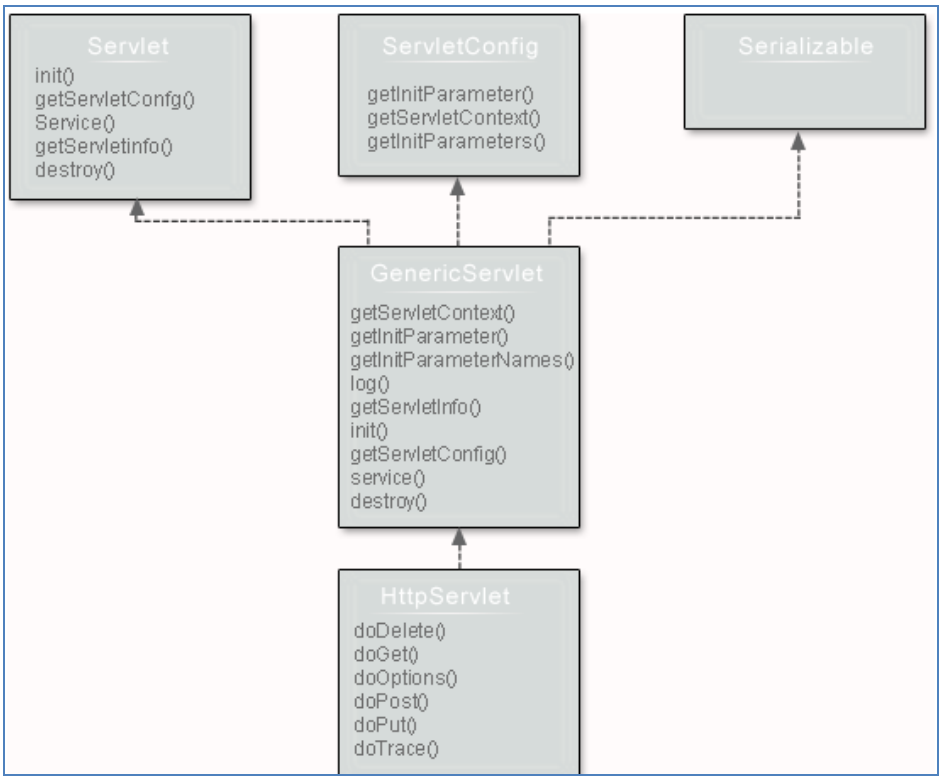
1. javax.servlet

Interfaces

1. Servlet – Declares life cycle methods for a servlet.
2. ServletConfig – To get initialization parameters
3. ServletContext- To log events and access information
4. ServletRequest- To read data from a client request
5. ServletResponse – To write data from client response

Classes

6. GenericServlet – Implements Servlet and ServletConfig
7. ServletInputStream – Provides an input stream for reading client requests.
8. ServletOutputStream - Provides an output stream for writing responses to a client.
9. ServletException – Indicates servlet error occurred.
10. UnavailableException - Indicates servlet is unavailable



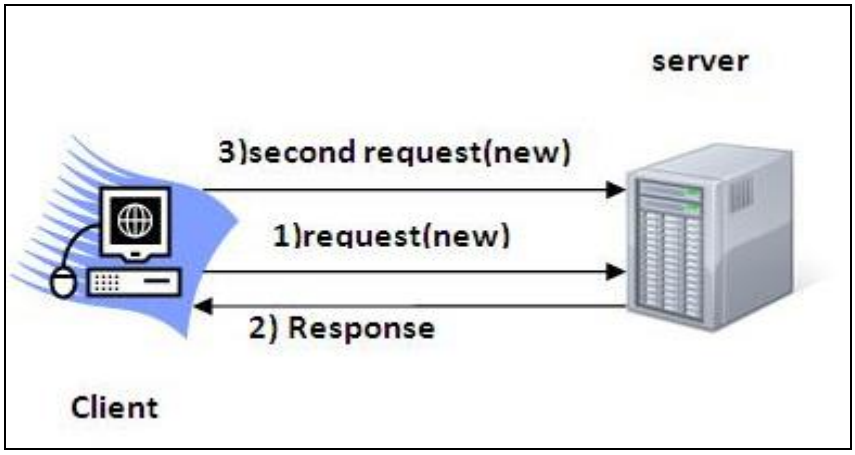
Servlet Interface

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

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

Session Tracking

- Session simply means a particular interval of time.
- Session Tracking is a way to maintain state (data) of an user.
- Http protocol is a stateless, each request is considered as the new request, so we need to maintain state using session tracking techniques.
- Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.

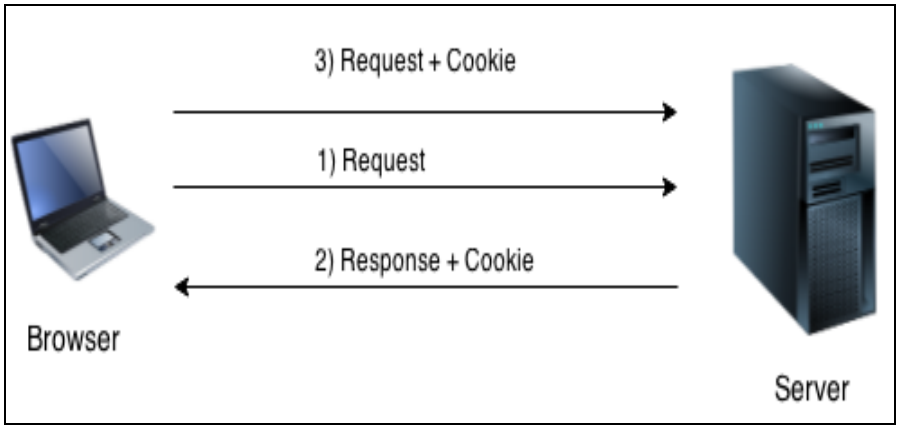


- We use session tracking to recognize the user It is used to recognize the particular user.

- Session Tracking Techniques
    - Cookies
    - Hidden Form Field
    - URL Rewriting
    - HttpSession

Cookies

- Cookies are text files stored on the client computer and they are kept for various information tracking purpose
- There are three steps involved in identifying returning users:
  - Server script sends a set of cookies to the browser in response header.
  - Browser stores this information on local machine for future use.
  - When next time browser sends any request to web server then it sends those cookies information to the server in request header and server uses that information to identify the user.
- Cookies are created using **Cookie** class present in Servlet API.
- For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:
  - **public void addCookie(Cookie ck):**method of HttpServletResponse interface is used to add cookie in response object.
  - **public Cookie[] getCookies():**method of HttpServletRequest interface is used to return all the cookies from the browser.



Disadvantage of Cookies

a. It will not work if cookie is disabled from the browser.
b. Only textual information can be set in Cookie object.

UNIT-IV

DATABASE ACCESS:

What is JDBC Driver?

JDBC drivers implement the defined interfaces in the JDBC API, for interacting with your database server.
For example, using JDBC drivers enable you to open database connections and to interact with it by sending SQL or database commands then receiving results with Java.
The *Java.sql* package that ships with JDK, contains various classes with their behaviours defined and their actual implementations are done in third-party drivers. Third party vendors implements the *java.sql.Driver* interface in their database driver.

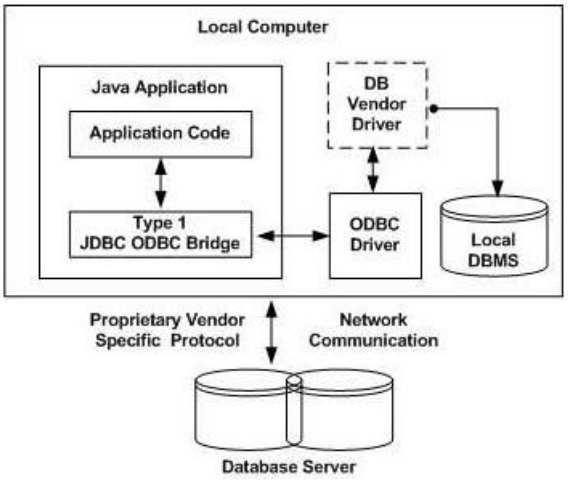
JDBC Drivers Types

JDBC driver implementations vary because of the wide variety of operating systems and hardware platforms in which Java operates. Sun has divided the implementation types into four categories, Types 1, 2, 3, and 4, which is explained below −

Type 1: JDBC-ODBC Bridge Driver

In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC, requires configuring on your system a Data Source Name (DSN) that represents the target database.

When Java first came out, this was a useful driver because most databases only supported ODBC access but now this type of driver is recommended only for experimental use or when no other alternative is available.

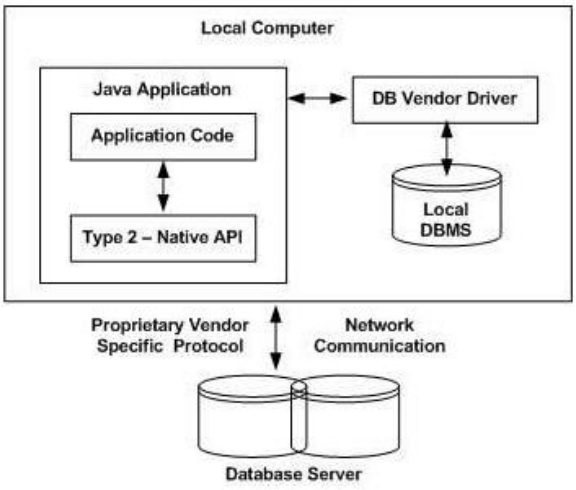


The JDBC-ODBC Bridge that comes with JDK 1.2 is a good example of this kind of driver.

Type 2: JDBC-Native API

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

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

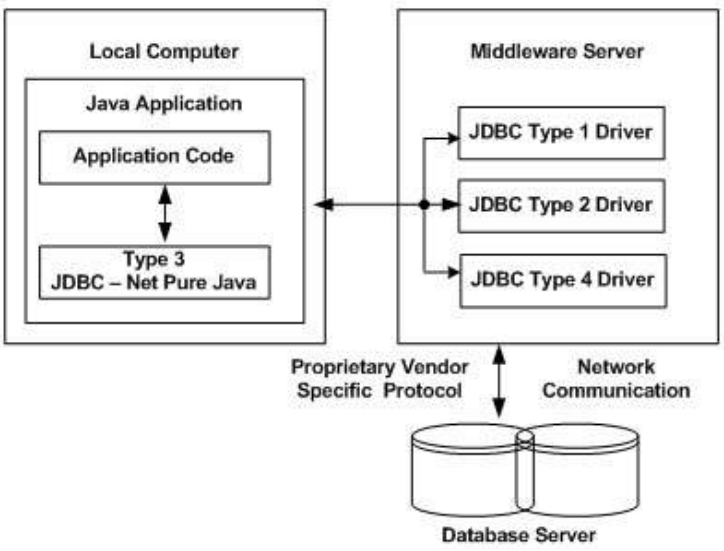


The Oracle Call Interface (OCI) driver is an example of a Type 2 driver.

Type 3: JDBC-Net pure Java

In a Type 3 driver, a three-tier approach is used to access databases. The JDBC clients use standard network sockets to communicate with a middleware application server. The socket

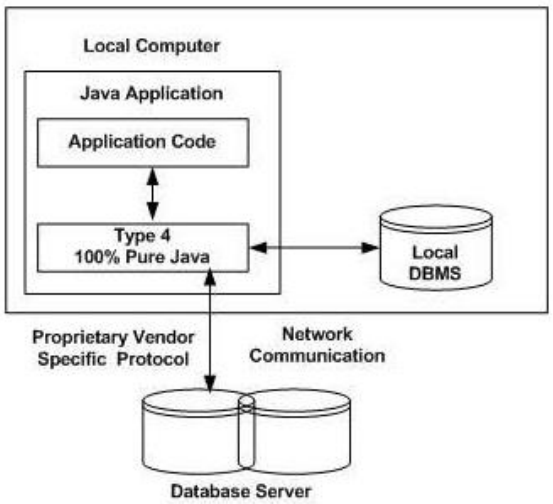
information is then translated by the middleware application server into the call format required by the DBMS, and forwarded to the database server.
This kind of driver is extremely flexible, since it requires no code installed on the client and a single driver can actually provide access to multiple databases.



You can think of the application server as a JDBC "proxy," meaning that it makes calls for the client application. As a result, you need some knowledge of the application server's configuration in order to effectively use this driver type.
Your application server might use a Type 1, 2, or 4 driver to communicate with the database, understanding the nuances will prove helpful.

Type 4: 100% Pure Java

In a Type 4 driver, a pure Java-based driver communicates directly with the vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself.
This kind of driver is extremely flexible, you don't need to install special software on the client or server. Further, these drivers can be downloaded dynamically.



MySQL's Connector/J driver is a Type 4 driver. Because of the proprietary nature of their network protocols, database vendors usually supply type 4 drivers.

Which Driver should be Used?

If you are accessing one type of database, such as Oracle, Sybase, or IBM, the preferred driver type is 4.
If your Java application is accessing multiple types of databases at the same time, type 3 is the preferred driver.

Type 2 drivers are useful in situations, where a type 3 or type 4 driver is not available yet for your database.
The type 1 driver is not considered a deployment-level driver, and is typically used for development and testing purposes only.

JDBC (Java Database Connectivity):

The first thing you need to do is check that you are set up properly. This involves the following steps:

1. Install Java and JDBC on your machine.

To install both the Java *tm* platform and the JDBC API, simply follow the instructions for downloading the latest release of the JDK *tm* (Java Development Kit *tm*). When you download the JDK, you will get JDBC as well.

2. Install a driver on your machine.

Your driver should include instructions for installing it. For JDBC drivers written for specific DBMSs, installation consists of just copying the driver onto your machine; there is no special configuration needed.
The JDBC-ODBC Bridge driver is not quite as easy to set up. If you download JDK, you will automatically get the JDBC-ODBC Bridge driver, which does not itself require any special configuration. ODBC, however, does. If you do not already have ODBC on your machine, you will need to see your ODBC driver vendor for information on installation and configuration.

3. Install your DBMS if needed.

If you do not already have a DBMS installed, you will need to follow the vendor's instructions for installation. Most users will have a DBMS installed and will be working with an established database.

Configuring Database:

Configuring a database is not at all difficult, but it requires special permissions and is normally done by a database administrator.
First, open the control panel. You might find "Administrative tools" select it, again you may find shortcut for "Data Sources (ODBC)". When you open the “Data Source (ODBC)" 32bit ODBC” icon, you’ll see a "ODBC Data Source Administrator" dialog window with a number of tabs, including “User DSN,” “System DSN,” “File DSN,” etc., in which “DSN” means “Data Source Name.” Select “System DSN,”. and add a new entry there, Select appropriate driver for the data source or directory where database lives. You can name the entry anything you want, assume here we are giving our data source name as "MySource".

JDBC Database Access

JDBC was designed to keep simple things simple. This means that the JDBC API makes everyday database tasks, like simple SELECT statements, very easy.

Import a package java.sql.* : This package provides you set of all classes that enables a network interface between the front end and back end database.

- DriverManager will create a Connection object.
- java.sql.Connection interface represents a connection with a specific database. Methods of connection is close(), creatStatement(), prepareStatement(), commit(), close() and prepareCall()
- Statement interface used to interact with database via the execution of SQL statements. Methods of this interface are executeQuery(), executeUpdate(), execute() and getResultSet().
- A ResultSet is returned when you execute an SQL statement. It maintains a pointer to a row within the tablur results. Mehods of this interface are next(), getBoolean(), getByte(), getDouble(), getString() close() and getInt().

Establishing a Connection

The first thing you need to do is establish a connection with the DBMS you want to use. This involves two steps: (1) loading the driver and (2) making the connection.

Loading Drivers**:** Loading the driver or drivers you want to use is very simple and involves just one line of code. If, for example, you want to use the JDBC-ODBC Bridge driver, the following code will load it

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Your driver documentation will give you the class name to use. For instance, if the class name is jdbc.DriverXYZ , you would load the driver with the following line of code:

```
Class.forName("jdbc.DriverXYZ");
```

Making the Connection: The second step in establishing a connection is to have the appropriate driver connect to the DBMS. The following line of code illustrates the general idea:

```
Connection con = DriverManager.getConnection(url,"myLogin", "myPassword");
```

If you are using the JDBC-ODBC Bridge driver, the JDBC URL will start with jdbc:odbc: . The rest of the URL is generally your data source name or database system. So, if you are using ODBC to access an ODBC data source called "MySource, " for example, your JDBC URL could be jdbc:odbc:MySource . In place of " myLogin " you put the name you use to log in to the DBMS; in place of " myPassword " you put your password for the DBMS. So if you log in to your DBMS with a login name of " scott " and a password of "tiger" just these two lines of code will establish a connection:

```
String url = "jdbc:odbc:MySource";
Connection con = DriverManager.getConnection(url, "scott", "tiger");
```

The connection returned by the method DriverManager.getConnection is an open connection you can use to create JDBC statements that pass your SQL statements to the DBMS. In the previous example, con is an open connection, and we will use it in the dorth coming examples.

Creating JDBC Statements

A Statement object is what sends your SQL statement to the DBMS. You simply create a Statement object and then execute it, supplying the appropriate execute method with the SQL statement you want to send. For a SELECT statement, the method to use is executeQuery . For statements that create or modify tables, the method to use is executeUpdate .

It takes an instance of an active connection to create a Statement object. In the following example, we use our Connection object con to create the Statement object stmt :

```
Statement stmt = con.createStatement();
```

At this point stmt exists, but it does not have an SQL statement to pass on to the DBMS. We need to supply that to the method we use to execute stmt . For example, in the following code fragment, we supply executeUpdate with the SQL statement from the example above:

```
stmt.executeUpdate("CREATE TABLE STUDENT " +
                   "(S_NAME VARCHAR(32), S_ID INTEGER, COURSE VARCHAR2(10), YEAR
                                        VARCHAR2(3)");
```

Since the SQL statement will not quite fit on one line on the page, we have split it into two strings concatenated by a plus sign (+) so that it will compile. Executing Statements

Statements that create a table, alter a table, or drop a table are all examples of DDL statements and are executed with the method executeUpdate. The method executeUpdate is also used to execute SQL statements that update a table. In practice, executeUpdate is used far more often to update tables than it is to create them because a table is created once but may be updated many times.

The method used most often for executing SQL statements is executeQuery . This method is used to execute

SELECT statements, which comprise the vast majority of SQL statements.

Entering Data into a Table

We have shown how to create the table STUDENT by specifying the names of the columns and the data types to be stored in those columns, but this only sets up the structure of the table. The table does not yet contain any data. We will enter our data into the table one row at a time, supplying the information to be stored in each column of that row. Note that the values to be inserted into the columns are listed in the same order that the columns were declared when the table was created, which is the default order.

The following code inserts one row of data, Statement stmt = con.createStatement();

```
stmt.executeUpdate( "INSERT INTO STUDENT VALUES ('xStudent', 501, ‘ B.Tech’,’IV’)");
```

Note that we use single quotation marks around the student name because it is nested within double quotation marks. For most DBMSs, the general rule is to alternate double quotation marks and single quotation marks to indicate nesting.

The code that follows inserts a second row into the table STUDENT . Note that we can just reuse the
Statement object stmt rather than having to create a new one for each execution.

```
stmt.executeUpdate("INSERT INTO STUDENT " + "VALUES ('yStudent', 502, ‘B.Tech’.’III’)");
```

Getting Data from a Table

Now that the table STUDENT has values in it, we can write a SELECT statement to access those values. The star (*) in the following SQL statement indicates that all columns should be selected. Since there is no WHERE clause to narrow down the rows from which to select, the following SQL statement selects the whole table:

```
SQL> SELECT * FROM STUDENT;
```

Retrieving Values from Result Sets

We now show how you send the above SELECT statements from a program written in the Java programming language and how you get the results we showed.

JDBC returns results in a ResultSet object, so we need to declare an instance of the class ResultSet to hold our results. The following code demonstrates declaring the ResultSet object rs and assigning the results of our earlier query to it:

```
ResultSet rs = stmt.executeQuery( "SELECT S_NAME, YEAR FROM STUDENT");
```

The following code accesses the values stored in the current row of rs. Each time the method next is invoked, the next row becomes the current row, and the loop continues until there are no more rows in rs .

```
String query = "SELECT COF_NAME, PRICE FROM STUDENT"; ResultSet rs = stmt.executeQuery(query);
while (rs.next())
{
String s = rs.getString("S_NAME");
Integer  i = rs.getInt("S_ID");
String c = rs.getString("COURSE");
String y = rs.getString(“YEAR”);
System.out.println(i  + "  " + s + "  " + c + "  " + y);
}
```

Updating Tables

Suppose that after a period of time we want update the YEAR column in the table STUDENT. The SQL
statement to update one row might look like this:

```
String updateString = "UPDATE STUDENT " +
"SET YEAR = IV WHERE S-NAME LIKE 'yStudent'";
```

Using the Statement object stmt , this JDBC code executes the SQL statement contained in updateString :

```
stmt.executeUpdate(updateString);
```

Using try and catch Blocks:

Java requires that when a method throws an exception, there be some mechanism to handle it. Generally a catch block will catch the exception and specify what happens (which you may choose to be nothing). In the sample code, we use two try blocks and two catch blocks. The first try block contains the method Class.forName, from the java.lang package. This method throws a ClassNotFoundException, so the catch block immediately following it deals with that exception. The second try block contains JDBC methods, which all throw SQLExceptions, so one catch block at the end of the application can handle all of the rest of the exceptions that might be thrown because they will all be SQLException objects.

Retrieving Exceptions

JDBC lets you see the warnings and exceptions generated by your DBMS and by the Java compiler. To see exceptions, you can have a catch block print them out. For example, the following two catch blocks from the

sample code print out a message explaining the exception:

```
try
{
// Code that could generate an exception goes here.
// If an exception is generated, the catch block below
// will print out information about it.
}
catch(SQLException ex)
{
System.err.println("SQLException: " + ex.getMessage());
```

Introduction to JSP: The Problem with Servlet. The Anatomy of a JSP Page, JSP Processing. JSP Application Design with MVC Setting Up and JSP Environment, JSP Declarations, Directives, Expressions, Code Snipplets, implement objects, Requests, Using Cookies and Session for Session Tracking.

The Servlet technology and JavaServer Pages (JSP) are the two main technologies for developing java Web applications. When first introduced by Sun Microsystems in 1996, the Servlet technology was considered superior to the reigning Common Gateway Interface (CGI) because servlets stay in memory after they service the first requests. Subsequent requests for the same servlet do not require instantiation of the servlet's class therefore enabling better response time.

Servlets are Java classes that implement the javax.servlet.Servlet interface. They are compiled and deployed in the web server. The problem with servlets is that you embed HTML in Java code. If you want to modify the cosmetic look of the page or you want to modify the structure of the page, you have to change code. Generally speaking, this is left to the better hands (and brains) of a web page designer and not to a Java developer.

```
PrintWriter pw = response.getWriter();

pw.println("<html><head><title>Testing</title

></head>"); pw.println("<body bgcolor=\"#

ffdddd\">");
```

As seen from the example above this method presents several difficulties to the web developer:

1. The code for a servlet becomes difficult to understand for the programmer.
2. The HTML content of such a page is difficult if not impossible for a web designer to understand or design.
3. This is hard to program and even small changes in the presentation, such as the page's background color, will require the servlet to be recompiled. Any changes in the HTML content require the rebuilding of the whole servlet.
4. It's hard to take advantage of web-page development tools when designing the application interface. If such tools are used to develop the web page layout, the generated HTML must then be manually embedded into the servlet code, a process which is time consuming, error prone, and extremely boring.
5. In many Java servlet-based applications, processing the request and generating the response are both handled by a single servlet class.
6. The servlet contains request processing and business logic (implemented by methods), and also generates the response HTML code, are embedded directly in the servlet code.

JSP solves these problems by giving a way to include java code into an HTML page using scriptlets. This way the HTML code remains intact and easily accessible to web designers, but the page can sill perform its task.

In late 1999, Sun Microsystems added a new element to the collection of Enterprise Java tools:

JavaServer Pages (JSP). JavaServer Pages are built on top of Java servlets and designed to increase the efficiency in which programmers, and even nonprogrammers, can create web content.

Instead of embedding HTML in the code, you place all static HTML in a JSP page, just as in a regular web page, and add a few JSP elements to generate the dynamic parts of the page. The request processing can remain the domain of the servlet, and the business logic can be handled by JavaBeans and EJB components.

A JSP page is handled differently compared to a servlet by the web server. When a servlet is deployed into a web server in compiled (bytecode) form, then a JSP page is deployed in its original, human-readable form.
When a user requests the specific page, the web server compiles the page into a servlet and from there on handles it as a standard servlet.

This accounts for a small delay, when a JSP page is first requested, but any subsequent requests benefit from the same speed effects that are associated with servlets.

The Problem with Servlet

- Servlets are difficult to code which are overcome in JSP. Other way, we can say, JSP is almost a **replacement** of Servlets, (by large, the better word is **extension** of Servlets), where coding decreases more than half.
- In Servlets, both **static code** and **dynamic code** are put together. In JSP, they are separated. For example,**In Servlets:**

  out.println("Hello Mr." + str + " you are great man");

  where **str** is the name of the client which changes for each client and is known as **dynamic content**. The strings, "Hello Mr." and "you are great man" are static content which is the same irrespective of client. In Servlets, in **println()**, both are put together.

- In JSP, the **static content** and **dynamic content** is separated. Static content is written in HTML and dynamic content in JSP. As much of the response comprises of static content (nearly 70%) only, the JSP file more looks as a HTML file.

- Programmer inserts, here and there, chunks of JSP code in a running HTML developed by Designer. As
  much of the response delivered to cleint by server comprises of static content (nearly 70%), the JSP file
  more looks like a HTML file. Other way we can say, JSP is nothting but Java in HTML (servlets are HTML
- in Java); java code embedded in HTML.
- When the roles of Designer and Programmer are nicely separated, the product development becomes cleaner and fast. Cost of developing Web site becomes cheaper as Designers are much paid less than Programmers, especially should be thought in the present competitive world.

- Both presentation layer and business logic layer put together in Servlets. In JSP, they can be separated with the usage of JavaBeans.
- The objects of PrintWriter, ServletConfig, ServletContext, HttpSession and RequestDispatcher etc. are created by the Programmer in Servlets and used. But in JSP, they are builtin and are known as "implicit objects". That is, in JSP, Programmer never creates these objects and straightaway use them as they are implicitly created and given by JSP container. This decreases lot of coding.

- JSP can easily be integrated with JavaBeans.
- JSP is much used in frameworks like Sturts etc.
- With JSP, Programmer can build custom tags that can be called in JavaBeans directly. Servlets do not have this advantage. Reusability increases with tag libraries and JavaBean etc.
- Writing alias name in <url-pattern> tag of web.xml is optional in JSP but mandatory in Servlets.
- A Servlet is simply a Java class with extension .java written in normal Java code.
- A Servlet is a Java class. It is written like a normal Java. JSP is comes with some elements that are easy to write.
- JSP needs no compilation by the Programmer. Programmer deploys directly a JSP source code file in server where as incase of Servlets, the Programmer compiles manually a Servlet file and deploys a .class file in server.
- JSP is so easy even a Web Designer can put small interactive code (not knowing much of Java) in static Web pages.
- First time when JSP is called it is compiled to a Servlet. Subsequent calls to the same JSP will call the same compiled servlet (instead of converting the JSP to servlet), Ofcourse, the JSP code would have not modified. This increases performance.

Anatomy of JSP

Once you have a JSP capable web-server or application server, you need to know the following information about it:

- Where to place the files
- How to access the files from your browser (with an http: prefix, not as file:)

You should be able to create a simple file, such as

```
<HTML>
<BODY>
 Hello, world
</BODY>
</HTML>
```

UNIT-V

JSP APPLICATION DEVELOPMENT:

GENERATING DYNAMIC CONTENT:

Dynamic contents means the contents that get changed based on the user inputs or states of external system or on some runtime conditions. JSP helps in handling such conditions. There are various ways by which JSP handles the dynamic contents such as use of:

1. Directive elements
2. Java Beans
3. Scripting elements
4. Standard tag libraries
5. Standard and custom actions

6. Expression language.

Directive Elements: Directive elements are used to specify the information about the page. Syntax: The directive names and attribute names are case sensitive. Examples of some directives are:

Page

- Include
- Taglib
- Attribute
- Tag
- Variable

JSP Scripting Element

JSP Scripting elements are written inside <% %> tags. These code inside <% %> tags are processed by the JSP engine during translation of the JSP page. Any other text in the JSP page is considered as HTML code or plain text.

Example:

```
<html>
   <head>
      <title>My First JSP Page</title>
   </head>
  <%
     int count = 0;
  %>
  <body>
       Page Count is  <% out.println(++count); %>
  </body>
```

Displaying Values Using an Expression to Set an Attribute

In all our JSP action element examples so far, the attributes are set to literal string values. But in many cases, the value of an attribute is not known when you write the JSP page; instead, the value must be calculated when the JSP page is requested. For situations like this, you can use a JSP expression as an attribute value. This is called a request-time attribute value. Here is an example of how this can be used to set an attribute of a fictitious log entry bean:

```
<jsp:useBean id="logEntry" class="com.foo.LogEntryBean" />
<jsp:setProperty name="logEntry" property="entryTime"
  value="<%= new java.util.Date( ) %>" />
```

This bean has a property named entryTime that holds a timestamp for a log entry, while other properties hold the information to be logged. To set the timestamp to the time when the JSP page is requested, a <jsp:setProperty>action with a request-time attribute value is used

Declaring Variables and Methods Error Handling

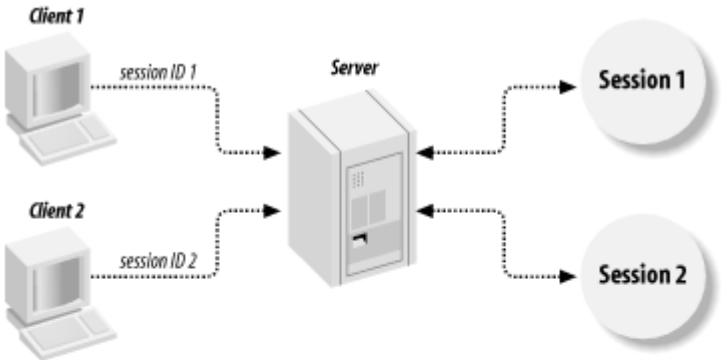
The exception is normally an object that is thrown at runtime. Exception Handling is the process to handle the runtime errors. There may occur exception any time in your web application. So handling exceptions is a safer side for the web developer. In JSP, there are two ways to perform exception handling:

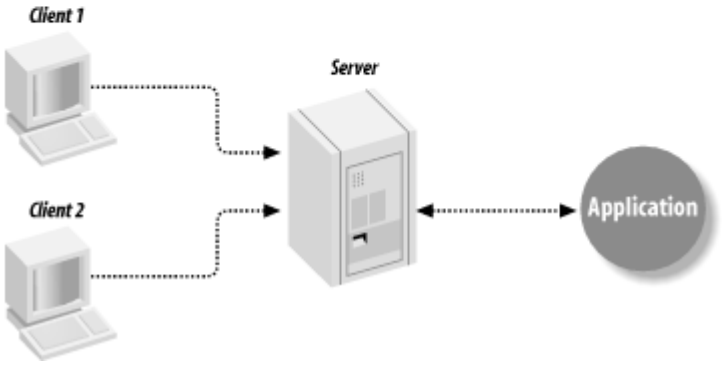
By errorPage and isErrorPage attributes of page directive
By <error-page> element in web.xml file
Example of exception handling in jsp by the elements of page directive
In this case, you must define and create a page to handle the exceptions, as in the error.jsp page. The pages where may occur exception, define the errorPage attribute of page directive, as in the process.jsp page.
There are 3 files:
index.jsp for input values
process.jsp for dividing the two numbers and displaying the result
error.jsp for handling the exception
index.jsp

```
<form action="process.jsp">
No1:<input type="text" name="n1" /><br/><br/>
No1:<input type="text" name="n2" /><br/><br/>
<input type="submit" value="divide"/>
</form>
```

Sharing Data Between JSP Pages, Requests, and Users

Any real application consists of more than a single page, and multiple pages often need access to the same information and server-side resources. When multiple pages process the same request (e.g., one page that retrieves the data the user asked for and another that displays it), there must be a way to pass data from one page to another. In an application in which the user is asked to provide information in multiple steps, such as an online shopping application, there must be a way to collect the information received with each request and get access to the complete set when the user is ready. Other information and resources need to be shared among multiple pages, requests, and all users. Examples are information about currently logged-in users, database connection pool objects, and cache objects to avoid frequent database lookups





Deploying Java Beans

A JavaBean is a specially constructed Java class written in the Java and coded according to the JavaBeans API specifications.

Following are the unique characteristics that distinguish a JavaBean from other Java classes −

It provides a default, no-argument constructor.

It should be serializable and that which can implement the Serializable interface.

It may have a number of properties which can be read or written.

It may have a number of "getter" and "setter" methods for the properties.

Introduction to struts framework

Struts is an open source framework that extends the Java Servlet API and employs a Model, View, Controller (MVC) architecture. It enables you to create maintainable, extensible, and flexible web applications based on standard technologies, such as JSP pages, JavaBeans, resource bundles, and XML.