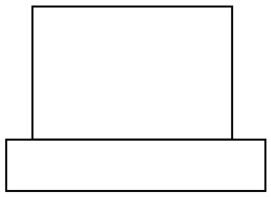
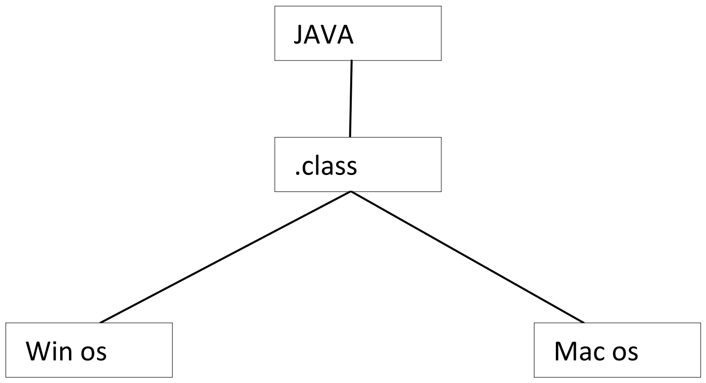
J2EE

Computer

- Computer is a combination of hardware and os.
- Graphically Computer are represented mention below:



- Application is a collection of programs with a dedicated functionality.
- Every application in this world is platform dependent <u>including java.</u>
- Java as an application is plantform dependent but as a programming language is platform independent.



- Every application will have its own known file extentions.
  Ex: pdf readers, media readers.
- Application acts like a interface between user and os.
- Os acts like a interface between application and <u>hardwear.</u>

user
App
Os
H/W

There are two types of applicaton:

1- Stand Alone or Desktop Application(Unshared Application)
2- Web Application (Shared Application)

Stand-Alone or Desktop Application:

i- These are the application which are present in our own computer and they are dedicated per user. For exapmple: pdf readers, media players.

Web Application:

i- These application are not present in our computer but they are presenet in "some other" computer and our computer and other (server) is network connected.

ii- In other words web application are present in networks.

iii- To interact with web application we must need to make use of
1- Network
2- Web Browser

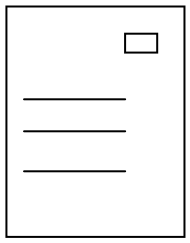
Networ is a collection of Computers and there are two types of network:

1- Intranet (Private)
2- Internet (Public)

- So web application can be present either in internet or intranet:
  For ex: Jspiders attendence tracking web application is present in intranet and gmail web application is present in internet.
- Whenever we make use of web browser it mean that we are interaction with web application.

Servers

Its also a computer but hardware of this computer is pretty high level compare to our own computers. For ex: Servers can have ram upto Gbs of rams…
Graphically server are represented as:



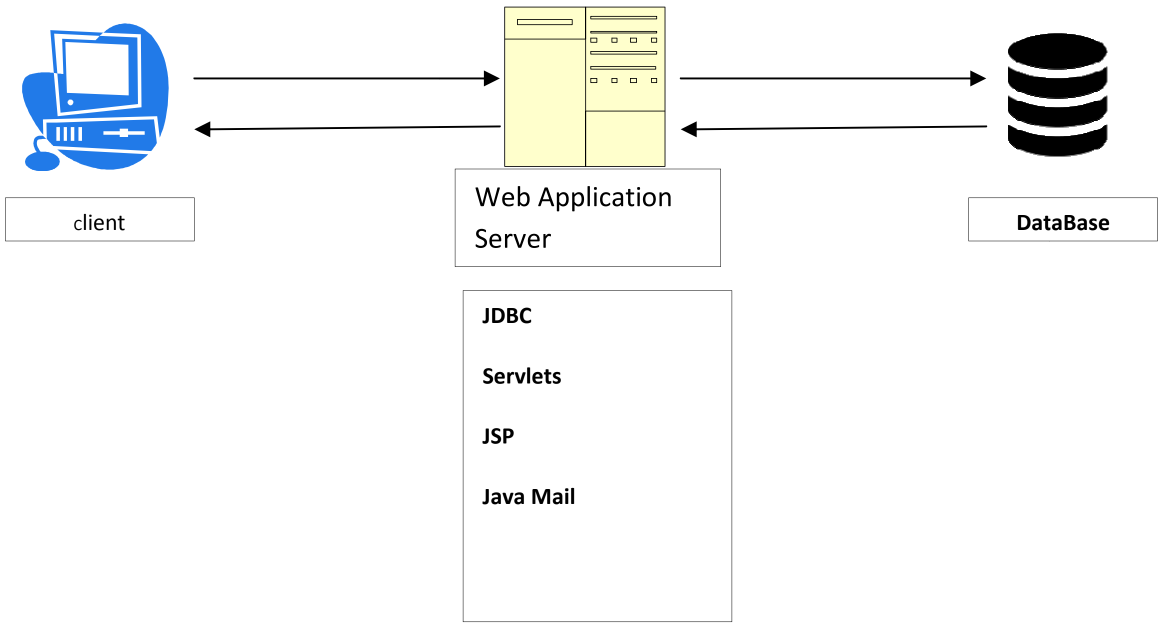
RDBMS Application (Database or DB)

It also an application which helps us to store and maintain gbs to tbs of data.

Graphically rdbms application is represented as :

J2EE: java is not enough to develop web application so we need some added featers …so J2EE WAS DEVLOPED.

J2EE: java 2 Enterprise Edition (it was introduced in java 1.2)



Packages are used to organize the java programs efficiently.(Collection of java Programs).

API- Collection of packages and every api has its own dedicated functionality.

In J2EE there are many APIs in J2EE like: JDBC, Servlets and JSP, JAVA mail, etc.

<u>Java 2 Enterprise Edition (J2EE, JEE, JAVA EE)</u>

J2ee is a collection of api which helps us to develop enterprise web application

An api (Application Programming Interface) is a collection of packages (One or More) with a dedicated functionality, Major API of J2EE are: JDBC , SERVLETS, JSP.

JDBC helps web application to interact to database

Servlets or jsp helps web application to get the request from browser, generates the response, and gives response back to the browser.

All the API's of J2EE are based upon java and hence they are platform independent.

Java DataBase Connectivity

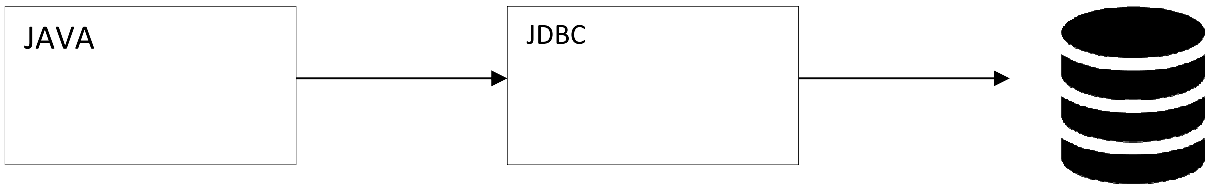
Java DataBase Connnectivity is an API, as the name implies, IT HELPS TO ACHIEVE THE CONNECTIVITY BETWEEN Java Programs & Database

Note: Servlets & JSPs are also java Programs

If we have a Web application & if it has a DB, then it needs to interact with DB to read/ modify the data

JDBC helps to do this & in the world of Java, JDBC is the "one and only" API that helps to interact with RDBMS (DB) Application.

Also JDBC is "DB independent" i.e. Using JDBC we can interact with any RDBMS Applications in the world.



JDBC Pre-Requirment:

i- Install Any RDBMS Appllication (MySQL)
ii- Create a "Database (Schema) " by any name.

iii- Create a table
iv- Insert some data into the table.

SQL Queries for MySQL RDBMS Application:-

1- create databse BECME89_DB;
2- use BECME89_DB;
3- create table students_info
```
(
regno Int(10) not null,
firstname varchar(50),
middlename varchar(50),
lastname varchar(50),
Primary key(regno)
);
```
4- insert into table_name values(regno,’firstname’,’middlename’,’lastname’);

Some UseFull Queries:

1- To Connect to database:
   use BECME89_DB;

2- To get the list of databases:
   show databases;
3- To get the list of tables:
   show tables;
4- To know the table structure:
   describe table_name;

Java ThumbRules :

A class which is declared with abstract keyword is called as abstract class.

A Class declare without abstract keyword is called as concrete class.

If LHS == RHS (A reg = new A();) then LHS Is always concrete class. If LHS != RHS , (B ref = new A();) then LHS can be an interface, concrete Class, or Abstract Class.

Simply having an abstract class is of no use and there must be at least one subclass.

Simply having an interface in java of no use and there must be at least one implementation class.

In java anything apart from primitive data types are called as object references. For example: Int I ,boolean isTrue(“i “ and “isTrue” is a primitive variable) , Object obj -> it’s an Object ref varialble.

Anything in java which starts with lower case is eithere variable, if its with brackets its a method. And Anything which starts with a Caps is either a Class or an Interface.

In Java, “Super Class” can be either an “Abstract Class” or “Concrete Class”.

Neccassary Steps to Work with JDBC

1- Loads the **Driver**
2- Get the **DB Connection** via **Driver**
3- Issue **SQLQueries** via **Connection**
4- **Processtheresults** returned by **SQLQueries**
5- Close All **JDBC Objects**

Note:-

>”java.sql.*” is the Package Representation of JDBC

> i.e Any Class / Interface belongs to this package means it’s part of JDBC

Drivers

Drivers are additional software components required by JDBC to interact with RDBMS applications. Drivers are provided by DB vendors and they are DB depentdent, that means, using mySql driver we can only interact with my Sql RDBMS application and using Db2 drivers we can only interact with Db2 RDBMS application.

JAR (Java Archieve) File:

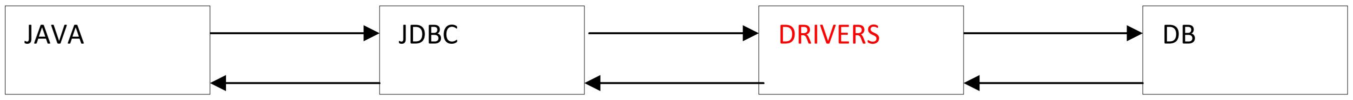
- It’s a collection of “.class” files + others Necessary Resources (Text File ,XML, Property Files, Etc.)
- JAR File helps us to transfer the “Java Files/ .class filse/ Java Application” From one location to an another location
- JAR File will have “.jar” file extension & Functionality wise it’s similar to “ZIP” file

Steps To Create JAR File:

1. Right Click on the Java Project, which we want to transfer , select “Export…”
2. Select “JAR file” option present under “Java” & click on “Next”
3. Provide the “Destination & File Name” , click on “Finish”

Steps To make use of JAR File:

1- Right Click on the Java Project, where we want to make use of JAR File, select “Build Path” click on “Add External Archieves…”
2- Select the “JAR File” & Click on “Open”
3- We see JAR File under “Referenced Libraries”



Driver Class

- "Driver Class" is a Concrete Class, present in driver JAR file, is the one that implements the "java.sql.Driver" interface
- This interface is present in JDBC API & every JDBC driver provider has to implement this Interface.
- "Driver" helps us to establish DB Connection, transfers the DB query and results between Java program and RDBMS Application.

<u>Steps to Load the "Driver Class" into the Program</u>

1. By invoking "registerDriver()" method present in "java.sql.DriverManager" Class by passing an instance of "Driver Class"
   Syntax:
   public vod DriverManage.registerDriver(java.sql.Driver driverRef) throws SQLException

for MySQL Driver:

com.mysql.jdbc.Driver ref = new com.mysql.jdbc.Driver();

DriverManager.registerDriver(ref);

Second approach

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

- This is the most common approach to resister a Driver Class which helps us to pass "Driver Class Name at Runtime"

Driver Types

There are 4 types of Drivers

1. Type 1: JDBC-ODBC Bridge
2. Type 2: Native-API Driver
3. Type 3: Network-Protocol Driver
4. Type 4: Native-Protocol Driver

Note: **In JDBC 4, the driver loads automatically, if the jar file is present in the project's Classpath.**

ODBC: Open Data Base Connectinvety

- DriverManager has 3 overloaded version of getConnection() methods

1. Connection getConnection(String dbUrl) throws SQLException

   ```
   String dbUrl =
   “jdbc:mysql://localhost:3306/BECEME89_DB?user=root&password=root”;
   Connection con = DriverManager.getConnection(dbUrl);
   ```

2. Connection getConnection(String dbUrl, String userNM,String password) throws SQLException

   ```
   String dbUrl =
               “jdbc:mysql://localhost:3306/BECME89_DM”;
   String userNM = “root”;
   String pass = “root”;

   Connection con = DriverManager.getConnection(dbUrl, userNM, pass);
   ```

3. Connection getConnection (String url, Properties info) throws SQLException

   ```
   String dbUrl = “jdbc:mysql://localhost:3306/BECEME89_DB”;
   String filePath = “”;
   FileReader reader = new FileReader(filePath);

   Properties props = new Properties();
   Props.load(reader);

   Connection con = DriverManager.getConnection(dbUrl, props);

   //Data Present in “db.properties” File is:-
   #DB Credentials
   User = root
   Password = root
   ```

   Note:

- We can make use of any version of “getConnection()” method to establish connection to RDBMS application
- But “getConnection(String url, Properties info)” helps us to take out the hardcoded credentials from program & keep it outside of the application
- Hence this method is widely used because it helps us to “easily maintain the application “ whenever there is change in DB credentials.

Static SQL queries: Queries which doesn’t have condition

Or

Condition values are hard Coded.

Dynamic SQL queries: There must be a condition plus one or more condition will be evaluated at the runtime. (?)

Static SQL Queries

- SQL queries
  - “Without conditions” OR
  - “with Hard Coded condition values” are called as “Static SQL Queries”

  Example:

  1- Select * from tablename;
  2- Create database DB_NAME;
  3- Select * from ABC where X = 1;

4- Insert into ABC values (1, 'Aatish');

Note: ABC = Table Name

Dynamic SQL Queries

- SQL Queries which
  - Must have conditions &
  - One/More condition values get decided at runtime are known as "Dynamic SQL Queries".

  Examples:

  1. Select * from ABC where X = ? and Y = ?;
  2. Select * from ABC where X = 1 and Y = ?;
  3. Insert into ABC values (?,"Praveen');

  Note:

  1. ABC = Table Name
  2. Dynamic SQL Query Must Contain One/More Question Marks.

JDBC Statements

- **JDBC** Statements send SQL queries to RDBMS and retrieve the data from RDBMS application.
- **There are different typed of JDBC Statements**
  1. java.sql.Statement
  2. java.sql.PreparedStatement
  3. java.sql.CallableStatement
- Once we create JDBC Statement Object (any of the above type) , then we MUST invoke any one of he below method to issue SQL queries to DB

1. Int executeUpdate() throws SQLExeception

- This method is used to execute "Other than SELECT " SQL queries.
- This method return "NO. of Rows Affected Count" in the form of Integer.

2. ResultSet executeQuery() throws SQLException

- ➢ This method is used to execute “ONLY SELECT” SQL Queries
- ➢ This method returns “DB Results” in the form of “ResultSet” Object

3. Boolean execute() throws SQLException

- ➢ This method is used to execute “ANY SQL Query including SELECT”
- ➢ This method:
  - Returns “true”,, if result is of type “DB Results”
  - Returns “false” , if result is of type “integer Count”
- ➢ If we use this method then we must make use
  - “getResultSet()”
    OR
  - “getUpdateCount()”
    Methods to get the actual results

java.sql.Statement

> Its an interface & an Object of Statement is used to execute “Static SQL Queries”

> Statement Object can be created by invoking “createStatement()” method on “Connection” Objects

Syntax:

Statement Connection.createStatement() throws SQLException

Statement stmt = con.createStatement();

Where “con” is the Object reference of “java.sql.Connection” Object

Q:Write a Java Program which deletes Reg. No. 6 data from “students_info” table;

Java.sql.PreparedStatement

- It’s an interface & an object of PreparedStatement is used to execute “Dynamic SQL Queries”
- PreparedStatement Object can be created by invoking “prepareStatement()” method on “Connection” Object.

  Syntax:
  PreparedStatement Connection.prepareStatement(String query) throws SQLException

Example:

String query = “delete * from students_info where regno = ?”;

PreparedStatement pstmt = con.prepareStatement(query);

Where “con” is the object reference of “java.sql.Connection” Object

- PreparedStatements MUST be used with query parameters (?) & these query parameters need to be set using proper setXXX () method before executing the dynamic SQL query

Syntax:

Void setXXX(Position of ? as Int Value, Corresponding Runtime Value) throws SQLException where XXX = Java Data Type corresponding to DB Column Data Type.

- PreparedStatements are also Known as “precompiled Statements” & they helps us to achieve “high performance”

Create “guardian_info” & “students_otherinfo” Tables in “database” eiht the following structure

“guardian_info”

1. Regno int(10) (pk)
2. Gfirstname varchar(50)
3. Gmiddlename varchar(50)
4. Glastname varchar(50);

Students_otherinfo

1. Regno int(10) (pk)
2. Isadmin varchar(1)
3. Password varchar(50)

Write a Java Program to insert regno 1 to 5 data into “guardian_info” table

Write a Java Program to insert regno 1 to 5 data into “students_otherinfo” table

While inserting data into “students_otherinflo” table make regno 1 “isadmin” value as “y” and rest as “N”

Stored Procedures

- Stored Procedures are group of SQL queries that performs a particular task (functionality wise they are similar to Java Methods)
- As its name implies, they are stored at RDBMS Application / DB side
- Stored Procedures helps to achieve “Reusability”
- Query to get the list of Procedures available in MySql Database is:
  SHOW PROCEDURE STATUS WHERE DB = DATABASE();

Stored Procedure 1:-

1- delimiter &
2- CREATE PROCEDURE getAllStudents()
BEGIN

SELECT * FROM students_info;

3- **END&**
4- **delimiter ;**

Stored Procedure 2:-

1- **delimiter $**
2- **CREATE PROCEDURE getStudentInfo(IN in_regno INT)**

BEGIN

SELECT * FROM students_info

WHERE REGNO = in_regno;

END$

3- **delimiter ;**
4- **call getStudentInfo (1);**

java.sql.CallableStatement:

- Its an interface & an Object of CallableStatement is used to execute "Stored Procedures"
- CallableStatement Object can be created by invoking "prepareCall()" method on "Connection" Object
  Syntax:
  CallableStatement Connection.prepareCall(String query) throws SQLException
  Example:
  String query = "call strotedProcedureNM()";
  CallableStatement ctmt = con.preparedCall(query);

  Where "con" is an Object reference of "java.sql.Connection Object"
- While invoking the Procedure, which takes input arguments
  -Either we can "hardcode the condition values"

Or

-These condition values may get decided at Runtime

> If condition values get decided at Runtime then we should have Question Mark(?) while constructing SQL Query

> Stored Procedures, by nature, reduces the number of DB calls

> Hence CallableStatements, which helps us to execute Stored Procedures, increases the "Performance of the Application"

Processing the Results returned by SQL Queries:

- **Whenever we issue SQL Queries to RDBMS Application via JDBC there are two kinds of results executed out of RDBMS Application**

1. **No. of Rows Affected Count**
2. **DB Results**

- **JDBC returns**
  - **No. of Rows Affected Count " As a "integer Value"**
  - **DB Results in the form of "ResultSet Object"**

java.sql.ResultSet

- **Its an interface & an Object of ResultSet is an "Object representation of DB Results" produced by Select SQL query.**
- **ResultSet object is produced by invoking "executeQuery()" Method on any of the JDBC Statements objects.**
- **ResultSet consists of N number of Rows with each row containing N number of Colum's.**
- **Number of rows and columns in ResultSet directly depends on "Where condition" & "column list" respectively in "Select Sql query"**
- **ResultSet object may consist of "Zero/More" Or "Zero/One" rows**
- **If resultSet consist of zero/more rows of data then we must use "while loop"**
- **If ResultSet consist of zero/one row of data then we can use either "while loop" or "if block" (preferred)**
- **Once the ResultSet is produced, data from ResultSet can be extracted as follow**

1. **Move to describe Row by calling necessary ResultSet methods:**
   For Ex: next(), first(),last(),etc
2. **Retrieve the desired column value using**

   getxxx(<Position of the Column in Sql Query as Integer Value>);

   Where xxx = java Data type corresponding to DB Table column data type

   Note: getxxx() methods are the ONLY way to retrieve data from ResultSet object.

<u>Closing Ceremony</u>

Why we need to Close Necessary JDBC Objects:

- JDBC Objects such as
  - Connection
  - JDBC Statements and

  - ResultSet
    Make use of memory
- In case of Connection Object, further RDBMS Application resources are cosumed
- Also memory consumed by ResultSet object is comparatively more compared to other JDBC objects
- Hence forgetting to close any of JDBC objects " will heavily impact the application performance" and "Garbage Collection" should not be relied upon
- So it's important to close any of the JDBC Object as soon as their job is done.
- To close any of the JDBC Objects invoke "close()" method

  Syntax:
  public void close() throws SQLException

Summary:

- While making use of JDBC we MUST follow 5 steps and out of 5, only once
  - We need to load the Driver (step 1)
  - We have to get the DB Connection (Step 2)
  - We have to Close JDBC Objects (Step 5)
- But ,Step 3 and 4 (i.e. Issuing SQL QUERIES & processing Results) can happen "N" number of times depending on our use
- Sets 1 to 4 will be in "try block" and step 5 will be in "finally block".
- Commonly used JDBC Objects are
  1. Java.sql.DriverManger
  2. Java.sql.Connection
  3. Java.sql.Statement
  4. Java.sql.PreparedStatement
  5. Java.sql.CallableStatement
  6. Java.sql.ResultSet
  7. Java.sql.SQLEcxeption
- Out of these apart from DriverManager & SQLException rest of them are "Interfaces". Where as DriverManager & SQLException are "Concrete Classes"
- SQLException is a Concrete Class which extends "java.lang.Exception" & it's a "Checked Exception"

Procedure 3

```
delimiter *
CREATE PROCEDURE StudentUpSert(IN in_regno INT,
                               IN in_fnm VARCHAR(50),
                               IN in_mnm VARCHAR(50),
                               IN in_lnm VARCHAR(50)
                               )
BEGIN
    DECLARE regno_count INT;

    SELECT COUNT(*)
    INTO regno_count
    FROM students_info
    WHERE regno = in_regno;

    IF regno_count>0 then
        UPDATE students_info
        set firstname = in_fnm,
                    middlename = in_mnm,
                    lastname = in_lnm
                WHERE regno = in_regno;
        ELSE
                INSERT INTO students_info
                VALUES (in_regno, in_fnm, in_mnm, in_lnm)
        END IF;
END*
delimiter ;
```

Write a java Program which accepts following input arguments in the same order

1. Reg no.
2. Current password
3. New Password

- This program first checks whether Reg no. & Current Password is matching
- If yes, Then Print the "Error Message" in Console
- If no, then print the "success Message" in Console along with updating the password for that reg no.

Servlets Introduction

Web Browser:

- It's a "Desktop Application" which helps us to interact with web applications

- Browser is the One & Only application which understands content/data present in HTML and display accordingly

Web Resources:

- Resources present inside a web application are called as web resources
- There are two types of web resources

  I- Static web Resources:

  - These resources "are present at web application" before making the request
  - Content of these resources "**does not change**".
  - In other words, resources which generates "static response" is called as Static web Resources
  - Few Examples:
    1. Any Songs Downloads
    2. Any Books (PDF, MS-WORD, etc.) Download
    3. Any Software Download
    4. Any Video/Movie files.

  II- Dynamic Web Resources:

  - These resources "**does not** present at web application" before making the request & they get generated at the time of request.
  - Content of these resources "May Change" for every request (Dynamic Response)
  - In other words, resources which generates "dynamic response" is called as Dynamic Web Resources
  - Few Examples:
    1. Any NetBanking Web Application Transaction Statement Download (PDF file)
    2. Any Post Paid Connection Statement Downloads (PDF file)
    3. Google Search Page (HTML Page)
    4. Gmail Inbox Page (HTML Page)
    5. Facebook Home Page (HTML Page)
    6. Gmail "Download All Attachments" (ZIP file)

    Note:

Both Static & Dynamic Web Response can be “HTML” or “Non-HTML” in nature.

Web Path:

- It’s a Path in Webserver in which Web Applications are present
- Web Path varies from Webserver to Webserver, we have to read the manual docs to get this information.
- In case of Tomcat web path is “<Tomcat Location>/webapps”
- Hence in Tomcat, WAR file should be kept webapps.

Starting the WebServer

Note:

- When we start webserver it should not throw any exception in the console
- At the time of starting the server, webserver extracts the contents of WAR file to a folder by same name inside “webapps” folder.

Web Application :

- Web Application is an application which is accessed over the network with the help of web browser
- Web application is a collection of web resources
- If a web application consists of “ONLY static resources then it is called as “Static Web Application”
- If a web application consists of “one/more dynamic resources” then it is called as “Dynamic Web Application”
  Example: Gmail, Facebook, Twiter, Flipkart, etc.,
- J2EE helps us to develop “Dynamic Web Applications”

Web Server:

- Like any other application (Adobe Reader, Media Player, etc.,) , Webserver is also an application which runs on Operating System
- Webserver as the name implies "Serves requests to a Web Applications"
- In other words, it helps both web browser & web application to interact with each other
- Hence every web application (Static/Dynamic) is directly under the control of webserver
- Few Examples:
  1- Apache Tomcat
  2- Apache JBOSS
  3- IBM WebSphere
  4- Oracle WebLogic
  5- Oracle GlassFish & many more…

Different ways to interact with Web Applications

1. By Typing an URL in Browser
2. By Clicking on the Hyperlink
3. By Submitting the HTML Form

WEB URL

- Web URL, uniquely identifies a particular web resource inside a web application
- Hence every web resource (Static/Dynamic) must need to have its unique address in the form of "web URL"
  Note: In case of Static web Resources, URL Consists of Resource file Name

Servlets:

- J2EE helps us to develop Dynamic web applications Hence Servlets & JSP acts like a Dynamic web Resources
- Servlets is an API of J2EE, it accepts web request from web server & generates "Dynamic Response" i.e response is getting generated at the time of Request.
- This Dynamic Response, may be a "HTML response" or "Non-HTML Response"
- For Example:
  1. Any NetBanking Web Application, Transaction Statement Download (PDF file) is a Non-HTML Response.
  2. Google Search Page (HTML Page) is a HTML Response
  3. Gmail "Download All Attachements" (ZIP File) is non-HTML Response
- In the world of Java, Servlets are the "One and Only API " that accepts web request and generate "Dynamic Response"
- Since Servlets are like Dynamic Resource & hence Servlets must have its unique address in the form of "Web URL"
- i.e Even though Servlets are Java Programs, we should not run them like normal Java Programs instead we should access via using corresponding Web URL with the help of Web Browser.

ASSIGNMENT:

Assume That there is a table by name "Library" With the following structure

"Library"

1. student_nm varchar(50)
2. book_nm varchar(50)
   NOTE: None of the above colums are Primary key

- Write a Java Program prints student name & corresponding number of books in the Console.

Steps to Create MY First Servlet:

1. Created the “Dynamic Web Project” by name “studentsApp” by selecting web module 3.0 or 3.1 to develop dynamic web Application

Note:

- In eclipse we should be in “Java EE” perspective to create Dynamic Web Project
- Dynamic Web Project has following 4 folders
  I- src (.java files will be present)
  II- Build (.class files will be present)
  III- WebContent (other than .java files will be present
- Create the “Index.html” file under “webContent” folder.
- Created the first servlet under “src” folder by extending “HttpServlet”
- Fixed the compilation error by copying the “servlet.jar” file to “WebContent/WEB-INF/lib” folder
- Override a method by name “doGet()”
- Javax.servlet.* is the package which represents servlet API
- In case of Dynamic Web Project, every JAR file SHOULD BE KEPT under “WebContent/WEB-INF/lib” folder
- Any Class which extends “javax.servlet.http.HttpServlet” is called as Servlet.

- Configure the URL for the servlet in “WebContent/WEB-INF/web.xml”
  Note: Every Servlet MUST have a unique URL & that URL MUST be present in web.xml.
- Generate the WAR File from Dynamic Web Project (Build Process)
  Note: To generate WAR file, right click on war file project, select “Export” & click on WAR file.

In the pop-up chose the destination to keep the WAR file.

WAR file "represents Dynamic Web Application" & helps to transfer dynamic web application from one location to another location

It's a Collection of .class files + Other necessary resources + Dependent JAR files (one/more).

2. Copied the WAR file to Web Server "Web Path" Location (Deployment Process)
   Web Path:

- It's a Path/Directory Location in Webserver in which Web Application are present.
- Web Path varies from Webserver to Webserver. We have to refer the manual to get this information
- In case of Apache Tomcat web path is "<Tomcat Location>webapps"
- Hence in Tomcat, WAR file SHOULD be kept under "webapps" folder.

3. Started the Webserver
   Note:

- When we start the webserver it should not throw any exception in the console
- At the time of starting the server, webserver extracts the contents of war file to the folder by same name inside webapps" folder.

4. Accessed the developed dynamic web application resources using Web Browser by using corresponding web URL's
5. Accessed the HTML Page (Static Web Resources)
   http://localhost:8080/appName/xyz.html

   NOTE:

   - In case of Static Resource web URL consists of "Resource File Nsme"
   - HTML generates "Static Response" or in other words it helps us to generate "Static Page

6. Accessed the Servlet by typing the configured URL present in web.xml (Dynamic Resource)
   http://localhost:8080/appName/servlet_public _url
   NOTE:
   - In case of Dynamic Resource web URL consists of “configured URL” present in web.xml
   - Servlet generates “Dynamic Response” or in other words it helps us to generate “Dynamic Page”.

Dynamic Web Resource = Dynamic Response = Dynamic Page

Static Web Resource = Static Response = Static Page.

We know that web application is a collection of web resources. Web URL uniquely identify these web resources inside a web application. Structure of web url is **:**
protocol://domain:port/path?query-string#fragment_id

Protocol:

- When one application wants to communicate with other (or in case browser & server) , there needs to be a common language which both application understands & that language should have set of rules and instructions
- “Protocol” is “Set of Rules”
- Web Browser & Web Server application communicate using
  1. Hyper Text Transfer Protocol (HTTP)
  2. Hyper Text Transfer Protocol Secure (HTTPS)
- As the name implies most of the time HTTP Response contain HTML

- In Url it's an optional informtion & default protoco is HTTP

Domain:

- IT uniquely identifies a computer in a network in which web application is present
- Domain consists of Computer Name / IP address of the computer in which web application is present
- In URL it is a Mandatory Information

Port:

- Port number in Web URL uniquely identifies web server application
- Default port number for http IS 80 & https is 443
- In URL this is an optional information
- When it's not used default port number is used depending on the protocol present in Web URL
- In Tomcat Webserver, default port number for HTTP is changed from 80 to 8080 and default port number for HTTPS is changed from 443 to 8443

Path:

- We know that web application is a collection of web resources (Static / Dynamic) & also Web Server can consist of one/more applications
- Path is the full path of the web resource at web application side
- It consists of Web Application Name + (File Name in case of Static Resource OR configred URL in case of Dynamic Resource)
- Web Application Name " uniquely identifies One web Application inside webserver.
- "File Name" uniquely identifies Static web resource inside that web application
- "Configured URL" uniquely identifies Dynamic web Resources
- In URL, it's an optional Information.

Query String:

- Query String is a name & value string pair which passes information ONLY to Dynamic Resources such as Servlets & JSPs
- In URL, It's an optional information and if present it starts with question mark followed by one or more name-value pair which are separated by an ampersnd(&)

Examples:
www.google.com/search?q=Knight

http://localhost:8080/studentsApp/currentDate?fname=xyz&lname=sfd

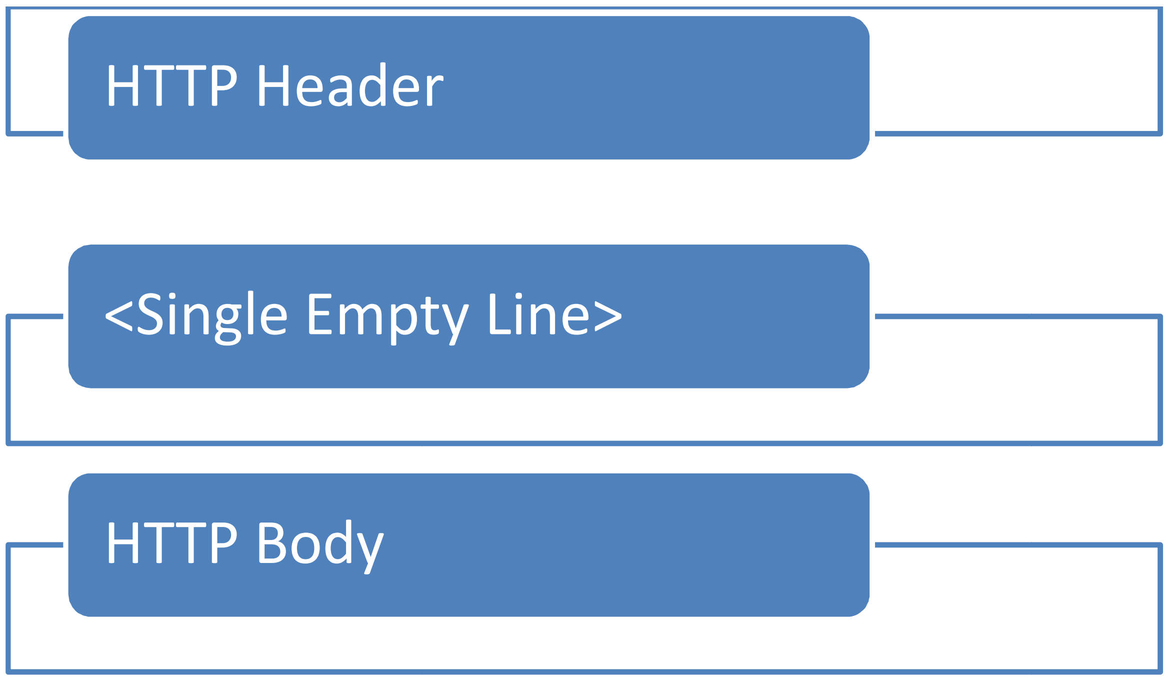
servlet Code to get Query String Information:-

String fnmVal = request.getParameter("Fname");
String lnmVal = request.getParameter("lname");

Where,
Request = Object reference of HttpServletRequests

Fragment ID

- A fragment ID or Fragment Identifier, as the name implies, it refers to a particular section within a web page
- In URL, It's optional information & if present, it begins with a hash (#) character followed by an identifier.
  Example:
  http://tomcat.apache.org/tomcat-6.0-doc/manager-howto.html#SessionStatistics



Key elements of HTTP Requests are:

1. URL
2. Form Data (if any)
3. HTTP Method
4. Cookies (if any)

Key elements of HTTP Response are:

1. Status code

2. Content Type
3. Actual Content
4. Cookies (if any)

Status Code:

- Status code represents the status of HTTP Request For example,

Status Code:	Description
200	Server successfully handled the request
404	Requested Resource (static/dynamic) is not found at server side
500	Server encountered an unexpected condition which prevented it from fulfilling the request.

- It's a Mandatory information & it will be present in Header of HTTP Response
- Generally Webserver provides "Status Code" info in Http Response

Content Type OR Multipurpose Internet Mail Extensions (MIME)

- Content Type OR Multipurpose Internet Mail Extensions (MIME) Type, tells the browser that what type of content it's going to receive so that it can prepare itself to handle the response data
- For example,
  - Open an Adobe Reader to handle PDF content
  - Open Media Player to handle media content etc.
- It's a mandatory information & it will be present in Header of HTTP Response

- The default content type is “text/html”
- Few Examples:
  text/html
  appplication/pdf
  video/quicktime
  Many more…

Actual Content:

- It’s a Mandatory information & it will be present in Body of HTTP Response
- In case of static resource content of the resource becomes the “Actual Content”
- In case of dynamic resource, content present in servlets/ JSP becomes the “Actual Content”
- In case of Error Scenarios webserver generates error information & it becomes the “Actual Content”.

Key Components Request:

Web Url

- Web application is a “Collection of Web Resources” & every web resource (static/dynamic) should have its unique address in the form of web URL
- Hence every request should consist of Web URL (Mandatory information) & it will be present in Header HTTP Request.

Form Data:

- Data collected using HTML form is called as Form Data
- i.e Whenever we make request by Submitting Form, Then ONLY HTTP Request will have Form Data
- Hence in HTTP Request it’s an Optional Information
- **If Present, it may be present in either Header or Body of HTTP Request which depends on HTTP Method Present in the Request.**

HTTP Method

- It’s a mandatory information present in the header of the HTTP Request
- HTTP Method is the first component in the HTTP Request header
- HTTP 1.0 had 3 mehods & in HTTP 1.1, Five new Methods got introduced so in total HTTP 1.1 has 8 different methods & every Http Request should consist of “ONE of the 8 HTTP Methods”
  (Present in HTTP 1.1)
  1. HEAD
  2. TRACE
  3. PUT
  4. DELETE
  5. OPTIONS
  6. GET
  7. POST
  8. CONNECT

  Code Word: htpp dog c
- Servlet API has “Default Implementation” for these methods (excluding CONNECT method)
- All these default implementations are present in the Servlet API Class by name “javax.servlet.http.HttpServlet”
- So whenever we create a Servlet by extending HttpServlet
  - We can inherit all these default implementations
    OR
  - We can override all of them
    OR
  - We can override couple of them
- **Depending on the HTTP method present in the request, Webserver invokes the corresponding doXXX(HSR, HSR) method**
- In other words, HTTP method indicate the desired action to be performed on the Dynamic Web Resource.
  i.e Servlet

Modifier and Type	Method and Description
`protected void`	**`doDelete`**`(HttpServletRequest req, HttpServletResponse resp)` Called by the server (via the `service` method) to allow a servlet to handle a DELETE request.
`protected void`	**`doGet`**`(HttpServletRequest req, HttpServletResponse resp)` Called by the server (via the `service` method) to allow a servlet to handle a GET request.
`protected void`	**`doHead`**`(HttpServletRequest req, HttpServletResponse resp)` Receives an HTTP HEAD request from the protected `service` method and handles the request.
`protected void`	**`doOptions`**`(HttpServletRequest req, HttpServletResponse resp)` Called by the server (via the `service` method) to allow a servlet to handle a OPTIONS request.
`protected void`	**`doPost`**`(HttpServletRequest req, HttpServletResponse resp)` Called by the server (via the `service` method) to allow a servlet to handle a POST request.
`protected void`	**`doPut`**`(HttpServletRequest req, HttpServletResponse resp)` Called by the server (via the `service` method) to allow a servlet to handle a PUT request.
`protected void`	**`doTrace`**`(HttpServletRequest req, HttpServletResponse resp)` Called by the server (via the `service` method) to allow a servlet to handle a TRACE request.
`protected long`	**`getLastModified`**`(HttpServletRequest req)` Returns the time the `HttpServletRequest` object was last modified, in milliseconds since midnight January 1, 1970 GMT.
`protected void`	**`service`**`(HttpServletRequest req, HttpServletResponse resp)` Receives standard HTTP requests from the public `service` method and dispatches

	them to the `do`*XXX* methods defined in this class.
`Void`	**`service`**`(ServletRequest req, ServletResponse res)` Dispatches client requests to the protected `service` method

POST Method

- If this method present in the Request then Webserver invokes doPost(HSR, HSR) Method in that Servlet
- This method allows user to Post the data of unlimited size to the server using HTML FORM (i.e Form Data)
- Post requests have a body
- Hence data sent using the POST is present in the body of the HTTP Request.

GET Method

- **If this method is present in the Request then Webserver invokes doGet(HSR, HSR) Method in that Servlet.**
- **This method allows us to get the data from server**
- **It's a default method**
- **Get requests "Using the GET Method the form-data is present in the Header part of HTTP Request "in the form of Query String"**

What determines whether Browser sends GET or POST requests:

1. **Typing a URL in Browser makes request to contain GET method**
2. **Clicking on a Hyper link in Browser makes request to contain GET Method**
3. **Submitting the form with method= "get" form attribute in Browser makes request to contain Get method.**

4. **Submitting the form with method = "post" form attribute in Browser makes request to contain "POST" method.**
5. **Submitting the form with "No method form attribute declaration " in Browser makes request to contain "GET" method.**

Note:

- Depending on the HTTP method present in the request corresponding doXXX()) method get executed at Servlet side
- If a form collects
  - "Sensitive Data" like Passowrd (ex: Gmail Login Page/ Profile Creation Page) OR
  - Very Large Data like sending mail (ex: Gmail Compose Mail option)
    Then that form should use method –"post"
- If a form collects "In-Sensitive Data" like search words (ex: Google Search) then that form may use method="post" or method="get"
- Hence whenever Servlet gets a request via Submitting the form then generally we override doPost() method & for rest of the cases, we have override doGet() method.

<u>Differences between GET/doGet() & POST/doPost()</u>

GET/ doGet	POST/ doPost()
GET method allows to get the data from server	POST method allows to Post data of unlimited size to the server.
GET is a default method	POST is not a default. We have to explicitly declare method ="POST" .
GET requests "do not have a Body" OR "have Empty Body"	POST requests "do have a body"
Hence incase of GET , Form Data will be present in HEADER IN THE FORM OF Query String.	In case of POST, Form Data will be present in Body.
Insecure; because form data get exposed to the outside world	Secure; because form data will be present in the Body & hence it will not be exposed to the outside world

The amount of data sent using GET is restricted because URL can contain Only limited characters.	There is no restriction on the amount of data sent using the POST method.
We cannot send the files using GET	We can send entire files using POST for Ex: Resume Upload, video files Etc.
GET requests, by default, they are “idempotent”. i.e we can perform the same operation again & again without any side effects.	POST requests are “Non-idempotent” in nature.
We “can bookmark” the GET requests	We “cannot bookmark” POST requests.

Servlet Container:

- Servlet Container is a sub-component of web server that helps both web server & servlet to communicate with each other.
- As the name implies, all Servlets of dynamic web application are directly under the control of Servlet Container.

HOW Servlet Container Works:

1. Whenever request comes, web server hand over the complete request to servlet container
2. Container by looking at the URL present in the request & referrring web.xml and then it comes to know the servlet which handles that request.
3. Container then “creates an instance” of that Servlet.
4. Once Instance creation is successful then it converts the “Raw HTTP Request” to a Java Object of type “HttpServletRequest” & also creates “HttpServletResponse” object.
5. Depending on the HTTP Method present in the request, container invokes corresponding doXXX() method by passing these request & response objects

6. doXXXX() method Once execution is over, container converts the response object to “Raw HTTP Response” & gives it back to web server
7. Once the response has been give back, Servlet Container garbage collects the request & response objects.
8. In other words for every request, container creates new request & response objects.
9. i.e the Life Span of these Objects is Created: once request comes to Servlet Destroyed: once response is given back .

Advantages of Servlet Container:

1. **Communication support:**
   Container helps both web server & servlet to communicate with each other.
2. **Multithreading Support:**
   Container automatically creates a new thread for every incoming request.
3. **Declarative Support**
   With “web.xml” which is used by Servlet container we can change the behavior of web application without changing anything in Servlet/JSP Code
4. **Life Cycle Management:**
   Container manages/controls the Life Cycle of a Servlet”
5. **JSP Supprt:**
   Container takes care of converting JSP into a Servlet.

Assingment: create a HTML Form as Name is : passwd: gender: education : tech language ; I have about me

Javax.servlet.http.HttpServletRequest

- "HttpServletRequest" object , in short called as "Request Object", is an Object representation of "Raw HTTP Request"
- We should make use of this object to get Information from Request.
- HttpServletRequest is an Interface & it extends another Interface by name "javax.servlet.ServletRequest".
- Request Object has many "Getter Methods" which helps us to get the information from "Raw Http Request"

Some Methods which are part of "Request Object" are:

1. **String HttpServletRequest.getMethod()**
   This methods returns the HTTP Method present in the request as a String Value
2. **StringBuffer HttpServletRequest.getRequestURL()**
   This method returns the URL present in the request as a StringBuffer.
3. **String ServletRequest.getProtocol()**
   This method returns the Protocol preseent in the request as a String Value.
4. geString ServletRequest.**getParameter**(String name)
5. String[] ServletRequest.**getParameterValues**(String name)

- Both the above Methods helps us to get the "Form Data/Query String information " form "Request Object"
- Both these methods return Null if the parameter name does not exist.

javax.servlet.http.HttpServletResponse

- "HttpServletResponse" Object, in short called as "Response Object", is an Object Representation of "Raw HTTP Response"
- We should make use of this object to send Info as part of "Raw HTTP Response"
- HttpServletResponse is an Interface & it extends an another Interface by name "javax.servlet.ServletResponse"
- Response Object has Methods which helps us to send the information as part of "Raw HTTP Response"

<u>Methods in Response Object:</u>

1. void ServletResponse.setContentType(String contentType)

- "setContentType()" Method present in Response object helps us to set the Content Type Info or Mime Type

2. PrintWriter ServletResponse.getWriter() throws IOException
3. void PrintWriter.println(String response)
   void PrintWriter.print(String response)

- These methods helps us to provide "Actual Content" info in Response Object
- First we should get "java.io.PrintWriter" from Response Object by invoking a method by name "getWriter()"
- PrintWriter has "print()/println()" methods which helps us to add Actual Content to Response Object.
  Note:
- PrintWriter is a "Concrete Class" but we SHOULD NOT create our own instance of this class , instead we Should get it from Response Object.
- Between "print()" & println()" methods print() reduces the size of the Actual Content there by increases the Performance.

4. void HttpServletResponse.sendError(int statusCode, string errMsg) throws IOException

- This method helps us to send the Error Response.

Assignments 2:

- **Create a HTML Form as CreateProfile.html**
- **Create Servlet by name "CrateProfileServlet" which gets the request from this form,**
  1. **Get the form data**
  2. **Strore the form data into corresponding Tables**
  3. **Generate Proper Response(Success/ Error Message)**

eXtensible Markup Language(XML) Introduction:-

- implies As the name it's an extension of HTML & this Language helps to Transfer or to store the Data between different Applications.
- XML looks similar to HTML but it's not a HTML Comparison between HTML & XML.

HTML	**XML**
1. **HTML** helps to display the Data in the Browser	1. XML helps to store and Transfer the Data between Application
2. **HTML has Pre-defined Tags**	2. **XML has User-defined Tags**
3. **HTML Tags are "Case In-Sensitive"**	3. **XML Tags are "Case Sensitive"**
4. **First Line of HTML is <!DOCTYPE html---->**	4. **First Line of XML is <?xml version="1.0" encoding="UTF-8"?>**
5. **HTML is "Not Strictly Typed" Language**	5. **XML is "Strictly Typed Language**
6. **File extension of HTML is ".html/.htm"**	6. **File extension of XML is ".xml"**

NOTE:

- Hence XML file should have
  - ".xml" as file extension
  - First line of that file should be
    <?xml version="1.0" encoding="UTF-8"?>
- **Since XML** CONSIST OF User-defined Tags, These tags information is defined in another file by name "XML Schema Document (XSD).
- **XSD** file will have ".xsd" as a file extension.
- **Hence** every application must obey the rules defined in XSD file while constructing the XML file & reading the data from XSD file.

<u>Deployment Descripter</u>

- **It's kind of "instruction sheet" to a Servlet Container & container always refer this to handle incoming requests**
- **It MUST**
  - **Be a XML file**
  - **Have the name "web.xml"**
  - **Be present inside WEB-INF folder.**
- **Hence every dynamic web application must have ONLY ONE web.xml.**

1. <welcome-file-list> Tag:-

- **This tag is used to configure default page for the web application**
- **If no resource name is specified in URL Path, then container searches the resources present in this tag in the order they have declared**
  Example:
  <welcome-file-list>

```
                <!—Static Resource-->
                <welcome-file>index.html</welcome-file>
                <!—Dynamic Resource -- >
                <welcome-file>currentDateTime</welcome-file>
    </welcome-file-list>
```

2. **Configuring a URL for a Servlet:**

- Every Servlet must have a URL & web.xml helps to configure a URL for a Servlet
- Container uses this information to identify a specific servlet to handle a given request
- There must be at least One Url configured for a Servlet. Also Servlet can have "more than One" URL
- Below are the different ways to configure the URL for a Servlet.

Below are the different ways to configure the URL for a Servlet

1. **Exact Matching**
2. **Directory Matching**
3. **Extension/Pattern Matching**

Example:

```
<!—I. Declaring the Servlet -- >

<servlet>

    <servlet-name>someName</servlet-name>

    <servlet-class>ffsa.jsdf</servlet-class>

</servlet>
```

<!—II. Configuring URL for a Serlet -- >

<! – 1. Exact Matching -- >

```
<servlet-mapping>
	<servlet-name> someName</servlet-name>
<url-pattern>/firstUrl</url-pattern>
<servlet-mapping>
```

<! – 1. Directory Matching -- >

```
<servlet-mapping>
	<servlet-name> someName</servlet-name>
<!-- <url-pattern>/abc/*</url-pattern> -- >
<url-pattern>/abc/SomeURL</url-pattern>
<servlet-mapping>
```

<! – 1. Pattern Matching -- >

```
<servlet-mapping>
	<servlet-name> someName</servlet-name>
```

```
<!-- <url-pattern>/abc/*</url-pattern> -- >

<url-pattern>*.do</url-pattern>

<servlet-mapping>
```

Order of Preference

1. Exact Match
2. Directory Match
3. Extension/Pattern Match

Servlet , ServletConfig

|

Generic Servlet

|

HttpServlet

|

MyServlet

javax.servlet.GenericServlet

- It's an abstract class, part of Servlet API a sub-class of GenericServlet is called as Servlet & it can handle "ANY Protocols" including HTTP & HTTPS protocols
- In other words , It becomes "Protocol –Independent Servlet"
- GenericServlet has "one abstract method" ,by name **service**(), hence it's an "abstract class"

  Syntax:

  public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
- Hence whenever we create Servlet by extending GenericServlet we MUST provide an implementation for service(SR,SR) method.

javax.servlet.http.HttpServlet

- A sub-class of HttpServlet is called as Servlet & it can handle ONLY HTTP & HTTPS protocols.
- In other words, It becomes "Protocol-Dependent Servlet"
- It's an abstract class but none of the methods in this class are declared as abstract.
- The implementation is
  - It checks whether request came via HTTP/ HTTPS protocol
  - If No, then it throws an ServletException with "non-HTTP request or response"
  - If YES, then it invokes "Overloaded version of service method i.e service(HSR, HSR)
- Implementation present in service(HSR, HSR) is
  - It gets the HTTP Method Present in the Request Object.
  - Invokes one of the corresponding doXXX(HSR,HSR) method by passing request response objects

- If request has HTTP Method as CONNECT, then this methods return "Method is not suppported by the Servlet API" error response.

➢ All the 7 doXXX(HSR,HSR) methods in HttpServlet has the logic of Generating "405 Error Response"

➢ A subclass of HttpServlet can override any of the below service() method
1. Pulbic void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
2. Public void service(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException
3. Protected void doXXX(HttpServletRequest req, HttpServletResponse res) throws ServletException , IOException

➢ We Should not override the first two versions of service methods & Our job is to override one/more doXXX(HSR,HSR) method(s)

➢ If we won't override doXXX() methods, then default implementation from HttpServlet is invoked which in turn return "405 Error response"

Why HttpServlet is an Abstract Class?

➢ HttpServlet does not have any abstract methods. But being "abstract", we are **forced to subclass**

➢ So Subclass can either
-inherit doXXX(HSR, HSR) methods OR
Override doXXX(HSR, HSR) methods.

➢ It's Servlet Developer choice & this choice is available "ONLY being an abstract Class with Zero methods as abstract"

➢ Thus, to force us to implement our own servlet class, the HttpServlet class is marked as abstract.

Difference between GenericServlet & HttpServlet

Generic Servlet	HttpServlet
Protocol Independent	Protocol Dependent. Supports only

	HTTP & HTTPS protocols.
Abstract Class; because service() method is declared as abstract	Abstract class; but none of the methods are declared as abstract
If we extend the GenericServlet then we must provide the implementation for service() method	There is NO restriction on overriding any version of the service method but generally we override one or more doXXX() methods.
GenericServlet does not extend any other Servlet API related class	HttpServlet extends GenericServlet which is part of Servlet API
GenericServlet implements Servlet API related interfaces such as "Servlet" & "ServletConfig"	HttpServlet does not implements any Servlet API related interfaces.

Assingment 4:

Create a HtML Form as shown below:

Login.html

Create a servlet by name "LoginServlet" which takes the request from this form,

1. Gets the Form Data
2. Authenticate the Credentials by interacting with BECME89_DB database.
3. If in-valid Credential, then display "in-Valid User Name/Password Error Message in Berowse
4. If Valid , then display corresponding Reg. No. Data (combination of Data Present in studentsinfo &guardian_info Tables), in the Browse as shown below.

Servlet LifeCycle:

Lifecycle of a Servlet is controlled by Servlet Container & it has following phases

1. Instantiation Phase
2. InitializationPhase
3. Service Phase
4. Destruction Phase

1. **Instantiation Phase:**

- Whenever request comes to a container, by looking at the Url and & referring web.xml container tries to find the Servlet name
- If No Servlet found, then it returns “404 Error Response”
- If Servlet found then Container creates an instance of the Servlet by invoking “Public Default Constructor ONLY”

2. **InitializationPhase**

Version 1:

```
public void init(ServletConfig config) throws ServletException
{
super.init(config)
//initialization code Goes Here
}
```

Version 2:

```
public void init() throws ServletException
```

```
{
//Initialization Code Goes Here
}
```

- After Successfully creating an instance, Container automatically invokes "init(ServletConfig)" Method
- Init(SC) method gives us a chance to initialize the Servlet before handling the requests. Like ,
  - Opening a Text File or
  - Reading the data from a Property File, etxl
- This method is called ONLY ONCE in servlet Lifecycle
- We may/maynot override this method. If we don't override then default implemetation present in GenericServlet is invoked
- The first line of the Version 1 init method should be "super.init(config)"
- **During initialization Servlet has access to following two key objects**
  1. **Javax.servlet.ServletContext**
  2. **Javax.servlet.ServletConfig**

Can we use Constructor for Initialization?

- We can also make use of Constructor for initialization but this approach is not so common
- Also , init method has access to ServletContext & ServletConfig objects where as Constructor don't

- So if initialization code is dependent on these two objects then we have to make us of init method. Also in case of constructor , we must make use of “Public Default Constructor”
- Hence we generally make use of init method for initialization.
- Once instantiation & initialization is successful, container caches the Servlet Instance

3 Service Phase:

public void service(ServletRequest req, ServletResponseres) throws ServletException, IOException

- After Instantiation & Initialization , Container creates Request & Response Objects, invokes service(SR, SR) method by passing these objects.
- This method is called “for every request” i.e one/more times in Servlet Lifecycle
- If a Servlet is a sub-class of GenericServlet then we MUST override its method
- If a Servlet is a sub-class of HttpServlet then we SHOULD NOT override this method & our job is to override one/more doXXX() methods.

4. **Destruction Phase**

public void destroy()

{

//Clean Up code Goes Here

}

- Whenever container wants to remove the cached Servlet Instance from it's memory then it invokes destroy() method "before removing"
- destroy() method gives us a chance to perform any clean-up activity such as Closing a File etc.
- This method is called ONLY once in Servlet LifeCycle
- We may/may not override this method. If we don't override then default implementation present in GenericServlet is invoked,

NOTE:

No matter how we create a Servlet, Container ALWAYS invokes below Lifecycle Methods on that Servlet

1. public default Constructor
2. void init(ServletConfig)
3. void service(ServletRequest, ServletResponse)
4. void destroy()

Note:

- Any Class which extends any one of the below class is called as a "Servlet"
  javax.servlet.HttpServlet
  javax.servlet.GenericSevlet

- **If** a class extends either HttpServlet or GenericServlet the "subclass of that class is also be called as Servlet".
- **Servlets (for which we configure a URL in web.xml) must be a "concrete class"otherwise they fail at runtime i.e. during the "instantiating Phase".**
- Servlets MUST have public default Constructor Or combination of any other constructor along with public default constructor.
- There is only one instance exist for any servlet. i.e. Servlets are "Singleton in nature".
- Servlets are protocol independent in nature but are most often used with HTTP & HTTPS protocols.
- At any point of time there will be multiple threads acting on servlet instance.
- **Hence by default servlets are Multi Threaded in nature. IN other words Dynamic Web Application are "Multi Threaded environments**"
  - **Marker interface has empty body.**
- **1. java.io.Seralizable**
  2 . java.lang.Cloneable
  3. Java.util.RandomAccess
  4. Java.rmi.Remote

  Servlet API

  1. Javax.servlet.SingleThreadModel.

<u>Single Threaded Servlets:</u>

- We know that , by default servlets are Multi Threaded in Nature. Hence following are the 2 ways to create Single Threaded servlet
  1. By Implementing "javax.servlet.SingleThreadModel" Marker Interface Or
  2. By Synchronizing Service Method.

- SingleThreadModel is a Marker Interface which ensure that servlets handle only one request at a time i.e. Container start handling the requests “Synchronously”
- This interface is “Deprecated” in Servlet API 2.4

  Example:

  ```
  public class MyServlet extends HttpServlet/GenericServlet implements
  SingleThreadModel
  {
  //Servlet code Goes Here
  }
  ```

Assignment 6:

>Create a HTML Page with Hyper-Link as Shown below

<u>AllStudentsView.html</u>

Create a Servlet which gets the request from this Hyper-Link & display All the students information(Combination_info table) & display the data in browser as shown below.

Diffrences between ServletContext & ServletConfig:

Servlet Context	**Servlet Config**
SevletContext is an Interface and “an Object of ServletContext” is used by container to pass information to ALL the servlets which are part of an	ServletConfig is an Interface and “an Object of ServletConfig” used by a container to pass information to a particular Servlet.

application	
ServletContext object is created at the time of Server Startup & Garbage collected during the server shutdown	ServletConfig Object is created during the “initialization Phase” of Servlet Lifecycle & Garbage collected during “Desstruction Phase”.
So there will be “ONLY ONE Instance of ServletContext object exists per web application”	There will be “ONLY ONE Instance of ServletConfig object exists per Servlet”
Hence “Singleton” in Nature	“Non-Sighleton” in Nature
ServletContext object is obtained by calling “getServletContext()” method which we inherited from GenericServlet	ServletConfig object is obtained by calling “getServletConfig()” method which we inherit from “GenericServlet”
ServletContext context = getServletContext();	ServletConfig config = getServletConfig();
In web.xml, context parameters are declared under <context-param> tag (one/more)	In web.xml, servlet config parameters are declared under <init-param> tag (one /more) which is a subtag of <servlet> tag.

ServletContext object “does not “ holds the object reference of ServletConfig	ServletConfig object holds the object reference of ServletContext
	
	ServletConfig config = getServletConfig();
	ServletContext context = config.getServletContext();

Note:

1. Both ServletContext & ServletConfig objects has a method by name getInitParameter() which helps us to get parameter value information from web.xml
   Syntax:

String getInitParameter(String paramName)

2. **We can ONLY get ServletContext & ServletConfig parameters at Runtime but "we can not set them"**

Assingment 7:

- Modify the response of Login Assignment in such a way that ONLY for Admin user display "Click Here" to View All Students info Hyper-link as shown below.

SendRedirect():

1. User makes the request to a servlet
2. Servlet redirects to another resource (Internal/External) . And Browser gets the URL as a response.
3. Borwser makes a New Request to this URL **When we redirect the request, it is always be a GET request.**
4. Browser displays whatever the response given by the new URL. Redirect happens at Browser side & In case of **Redirect URL in the browser changes.**
5. To redirect the request call sendRedirect() on the HttpServletResponse object.

Syntax:

void HttpServletResponse.sendRedirect(String url) throes IOException

Example:-

String url = null;

//Redirect – Internal URL (Dynamic)

url =” http://localhost:8080/studentsApp/currentDate;

url =”currentDateTimme”;

//Redirect- Interal URL –static Resource

url= http://localhost:8080/studentsApp/index.html;

url=”index.html”;

resp.sendRedirect(url);

Create a HTML Form as shown below

GraingerSearch.html

Create a Servlet which gets the request frm this form, takes the “keyword” & display the results for that keyword in www.grainger.com Website.

FORWARD:

1. User makes a request to a servlet

2. Servlet internally forwards that request to another servlet by passing Request & Response objects (i.e. forward happens at Server side), Another servlet handles that request & gives back the response.
3. Browser displays the response. In this case Browser will not have any clue on what went behind the scene. Also, URL in the browser doesn’t change.
4. To forward the request call forward() on the RequestDispatcher object.

void RequestDispatcher.forward(ServletRequest req, ServletResponse resp) thorws ServletException, IOException

- We can get the “RequestDispatcher” Object, by invoking “getRequestDispatcher()” Method on “Request” Object

Example:

```
RequestDispatcher

        ServletRequest.getRequestDispatcher(String url)

RequestDispatcher dispatcher = null;

String url = null;

url = “currentDateTime”;//Internal Resource – Dynamic

url = “index.html” ; //Internal Resource – Static

dispatcher = req.getRequestDispatcher(url);

dispatcher.forward(req,resp);
```

Note:

- We cannot forward the request to External Resources For Example:

```
String url = "http://www.google.com";
Dispatcher = req.getRequestDispatcher(url);
Dispatcher.forward(req,resp);
```

Diffrence between redirect and forward:-

Redirect	**Forward**
1. Redirect happens @ Browser side	1. Forward happens @ "Server side"
2. URL in the browser changes	2. URL does not change
3. We can Redirect the request to "both Internal & External Web Resources"	3. We can forward the request "ONLY to Internal Web Resources"
4. Redirect contains "More Than One" request & response cycle	4. Forward contains "ONLY ONE " request & response cycle
5. In case of Redirect, "more than one set of Request & Response Objects get created	5. In case of Forward, "ONLY ONE set of Request & Response Object" get Created
6. Slower in operation	6. Faster in operation
7. Redirect makes request to contain HTTP GET method & hence it ALWAYS invokes doGet() method at destination	7. When we forward the request, it will invoke the corresponding doXXX() method at destination,
8. Redirect happens on "Response Object"	8. Forward happens on "Request Object"

NOTE:

1. If URL in Browser Changes means it MUST be a Redirect (especially Domain Name)
2. For Internal URL either we can make use of Redirect / Forward but "Forward" is preferred
3. For External URL we left with no choice we MUST make use of Redirect In other words.
4. If one web application wants to communicate with another, then Redirect is the OnlY way , In this case both the web applications can transfer the data using "Query Parameters"
5. We cannot redirect/Forward the request to more than one URL at a time.

Include:

- RequestDispatcher is an interface which has following 2 abstract Methods
  1. public abstract void forward(ServletRequest req, ServletResponse resp) throws ServletException, IOException
  2. public abstract void include(ServletRequest req, servletResponse resp) throws ServletException , IOException
- An object of RequestDispatcher helps us to perform either Forward or Include Operation

- “forward() method” helps us to Forward the Request from One Servlet to any other Internal Resource(Static / Dynamic)
- Include() method “helps us to Include the Response of an another Internal Resource (Static/ Dynamic) into Servlet
- When we Include the content of one servlet into an another, it will include the response of “corresponding overridden version of doXXX() method “ in that servlet
- If corresponding overridden version of doXXX() methods does not exists, then Container “Does Not give any Complication Error/ Runtime Exception” In this case it just ignore the include statement.

  Example:

  ```
  RequestDispatcher dispatcher = null;
  Out.print(“1111111111”);
  ///Internal Resource –Static
  Dispatcher = req.getRequestDispatcher(“index.html”);
  Dispatcher.include(req, resp);
  Out.print(“2222222”);

  //Internal Resource – Dynamic
  String url = “currentDate?fname=Praveen&lname=D”;
  Dispatcher = req.getRequestDispatcher(url);
  Dispatcher.include(req, resp);
  Out.print(“3333333”);
  ```

  NOTE:
- We cannot include the response of “External Resource” into the Servlet

- Unlike Redirect & Forward , we can include more than one resource/URL at a time.
- Include helps us to reue the internal Resource with that its "easy to maintain the Web Application"

Assignment 8:

- Create a HTML Form as shown below
- Advancesearch.html
  Create a Servlet
  - Which gets the request from this form,
  - Takes the "keyword"
  - Display the results for that keyword in ther correcponfing Website.

<u>Assignment : 9</u>

- Create a "Header Page " as shown below
  Header.html
- Provide the "href" for
  - Creaate Profile
  - Search
  - Change Password
- Don't Provide "href" for "Home" & "Logout"
- Create a "Footer Page" as shown below
  Footer.html
- Include theseHeader.html & Footer.html in the responses of
  - Login Page
  - Create Profile

- Search
- Change Password

➢ Also, if Authentication failes durng Loogin, "Generate Login Page with Errot Message" as a Response as shown Below.

Cookies

➢ Cookies are little piece of information in the form of name-value string pair exchanged between browser & server

➢ Cookies are created by server and "Maintained by Browser" & hence Cookies are "Browser Dependent.

➢ Cookies are the one & Only way in Servlet API , if we want to store any information in "Browser".

➢ Browser before sending the request, it performs the below activities (automatically)

1. It takes "Domain Name" from request
2. Looks out for Cookies within it's own memory
3. If present, it attaches cookies to Request
4. If NOT present it simply sends the Request to Server without Cookies

➢ Hence Cookies "optional" both in HTTP Request & Response

➢ There are two typed of cookies,

1. Non-Persistent Cookie
2. Persistent Cookie.

1. Non-Persistance cookie (is default)

➢ Non-Persistent Cookie lives as long as Browser is kept open. Once the Browser is closed cookie disappears

➢ **Persistent Cookie**:

- Persistent Cookie will be present in Browser even after Browser is Closed.

<u>public Cookie Constructor(String name, String value)</u>

public Cookie(String name, String value)

- It's the ONE & ONLY constructor available in "javax.servlet.http.Cookie" Concrete Class
- It creates the Cookie Object with specified name & value
- Name can contain only "Alphanumeric Character" but should not contain comma, white space, semicolons.
- Cookies name "Cannot be changed " after creation
- Value can be anything & it "can be changed" after creation.

2. <u>Void HttpServletResponse.addCookie(Coookie cookieObj)</u>

- This method add the specified Cookie to the Respnse
- This method can be called "multiple Time" to set more than one Cookie to the Response

3. <u>Cookie[] HttpServletRequest.getCookies()</u>

- This method returns
  - Nan "Array containing all the Cookie Objects"
    Or
  - Return "NULL" if request does not have Cookies

4. **<u>String Cookie.getName()</u>**

- This method returns the Name of the Cookie

5. <u>String Cookie.getValue()</u>

- This method return the value of cookie

6. **void Cookie.setMaxAge(int expiry)**

- Set's the max age of Cookie in "Seconds"
- +ve value makes Cookie "Persistent"
- Any –ve Value makes Cookie "Non-Persistent
- 0 Value make Cookie to be "Deleted Immediately"

Example:

1. Create Coookie

```
//Non-Persistent Cookie
Cookie myNameCookie = new Cookie("myName","Praveen D");
//Persistent Cookie
Cookie myLocationCookie = new
Cookie("myLocation","Bangalore");
//Time in Seconds
myLocationCookie.setMaxAge(7*24*60*60);
//send the above Cookies in the Response
resp.addCookie (myNameCookie);
resp.addCookie (myLocationCookie);

out.print("Cookies Created…");
```

2. **Read Cookie**

```
//Get Cookies From Request
Cookie[] receivedCookies = req.getCookies();
If(receivedCookies==null){
out.print("Cookies are NOT Present");
}else{
    out.print("Cookies are present");
```

```
        for(Cookie rcvdCookie : receivedCookies){
        //Print the Cookie Info (Name & Value) in Browser
        out.print("Cookie Name:" +rcvdCookie.getName());
        out.print("Cookie Value: "+rcvdCookie.getValue());
        }
    }

}
```

3. **<u>Delete Cookie</u>**

```
Cookie[] receivedCookies = req.getCookies();
If(recievedCookie == null){
Out.print("Cookies are NOT present");
}else{
Out.print("Cookies are present");
for(Cookie rcvdCookie : recevedCookies){
String Name = rcvdCookie.getName();
//Delete ONLY 'myLocation' Cookie
If(name.equals("myLocation")){
rcvdCookie.setMaxAge(0);
resp.addCookie(rcvdCookie);
out.print("Deleted 'myLocation' Cookie");
break;
}
}
}
```

Assignment 9:

- Implement "Remember User Name" functionality in "studentsApp" web Application

  Note:

For this reqirement, Login Page should be Dynamic
Hecne we must stop using Login.html & create a Servlet (ec: LoginPageServlet) for generating Login Page

- We have to midify followeiing 2 servlets for this requirment
  1. LoginServlet
  2. LoginPageServlet

Attributes:

- Attributes helps us to pass information in the form of "name = value" pair where name is a
  String" & value is "java.lang.Object"
- Which means that, any java object can be an attribute value
- Attributes are the one and only way , in Servlet API, to pass information in the form of "java object" from one servlet/JSP to another servlet/JSP
- Attributes are of 3 types
  1. Context Attributes (Application Scope)
  2. Request Attributes (Request Scope)
  3. Session Attributes(Session Scope)
- These 3 attributes can be get / set by using following Objects respectively
  1. Javax.servlet.ServletContext
  2. Javax.servlet.ServletRequest
  3. Javax.servlet.http.HttpSession
- Following methods are present in above Three Objects, which can be used to "get, set or remove" attributes

1. **public void setAttribute(String name, Object value)**

- Sets an Object to a given attribute name
- If an attribute with the given name already exists then this method will replace the existing object with the new Object

2. **<u>Public Object getAttibute(String name)</u>**

- This method returns an attribute value as java.lang.Object with the given name OR returns NULL if there is no attribute by that name exists.

3. **<u>public void removeAttibute(String name)</u>**

- This method removes an Attribute with the given name
- After removal, subsequent calls to getAttribute() with the same name returns NULL

<u>Note:</u>

- Following are the different ways of passing information to Servlets
  1. Using Query String
  2. Using Context & Config parameters
  3. Using HTML form
  4. Using Cookies
  5. Attributes

- Apart from Attributes rest are name=value pair. In Attribute name is a "String" but value is an "Object"
- Hence if one Servlet wants to pass information to another in the form of "Java Object" then "Attributes" are the One and only way.

No.	ServletContext	ServletRequest	HttpSession
1. Instance	One Per	One per Request	One per User OR

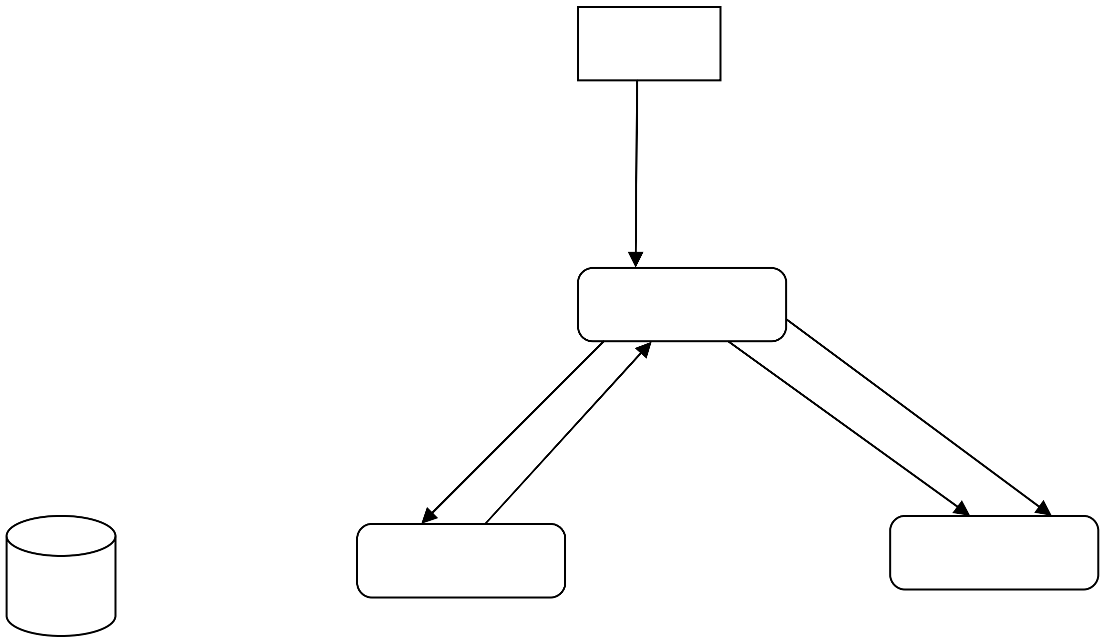
	Application	& Response Cycle	Login/Logout
2. LifeSpan	Created during server start up Destroyed during Server shutdown	Created when Request Comes Destroyed When Response is Given	Created When User Login Destroyed When User Logout
3. Scope	Application	Request & Response Cycle	Session

4. Functionality	All the Above 3 Objects has private HashMap<String, Object> & 3 Public Methods to Operate on this HashMap i.e. 1. Void setAttribute(String, Object) 2. Object getAttribute(String) 3. Void removeAttribute(String)

- It's also known as "Value Objects (vo) " Design pattern, helps us the transfer the data between one Java Program to Another in the form of Java Objext using "Java Beans or POJO"
- This way of passing the data is easy as compared to other ways like , Arrays, ArrayList, HashMap etx.
- Also this design pattern helps us to write "error free" data transfer code in loosely coupled fashion.

MVC Design Pattern:

-



- This design pattern divides entire request & response flow into three components & it defines the interactions between them
  1. Model
  2. View
  3. Controller

1. Model: (Data Model):-

- It's the core part of the application & consists of application business logic
- In J2EE application "plain Java Classes" are used as models

2. View:-

- The view captures the data from the user & also displays the data to the user
- It does not worry about what that data means, or where to store that data etc.
- In j2ee applications “HTML/ JSP’s” are used as Views.

3. Controller:-

- Controller controls both model & views. In other words it controls the entire request and response flow.
- It takes the request from View, calls the Model to get some data and passes the data to the vies for displaying to user

<u>MVC Design Pattern Advantages:-</u>

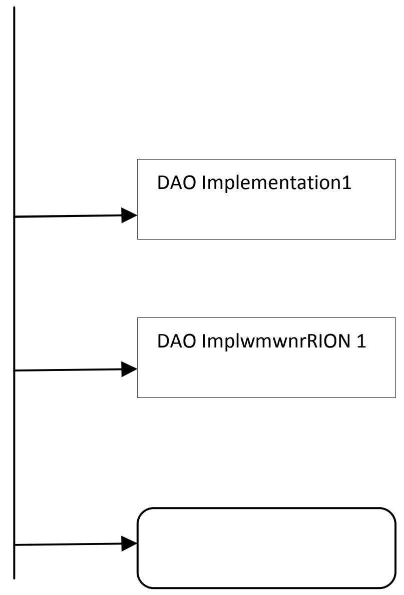
- MVC Design pattern helps us in achieving
  - Readability
  - **<u>Maintainability</u>**
  - Flexibility
  - Reusability
  - Scalability

  Of the Web Application
- Hence MVC Design pattern is most widely used Design Pattern to develop Web Applications
- For Example, Struts & Spring Frameworks internally uses MVC Design Pattern

<u>DAO Design Pattern:-</u>

DAO interface

DAO interface exposes only the relevant data access methods to other programs



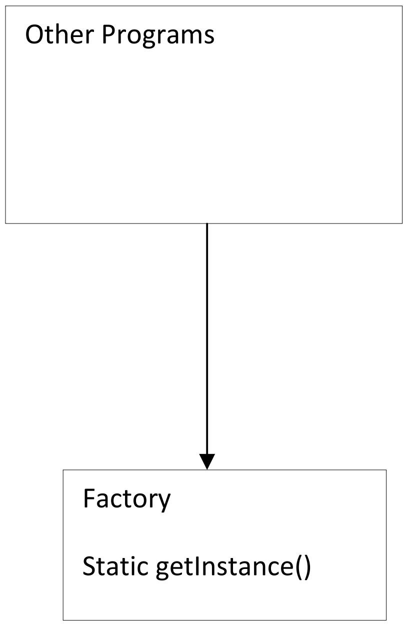
- The purpose of DAO's is to help us to interact with database
- This design pattern defines data access methods in an Interface & corresponding implementation logic will be present in a "Concrete Class" which impemets this interface
- Therefore we can hava "N" number of implementation class to align with different type of data access mechanism we want to use
- For examplw,
  - One implementation class uses JDBC
  - Other one uses Hibernamt
  - One more uses Spring-JDBC, etc..
- As shown in the diagram, the DAO interface exposes only the relevant data access methods to the other programs by hiding implemetaion detalis.
- With DAO design pattern it is quite easy to swap from one implementation to another with minimal impact on application

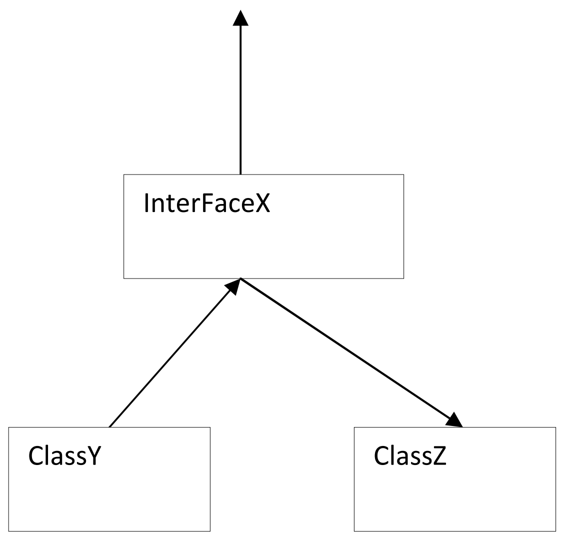
because the interface exposes by the DAO to other programs does not change when the underlying data handling logic changes.

Factory Design Pattern:

Other programs need not to own the responsibility of creating Class Y or Class Z object using new operator instead they get instance of either Class Y or Class Z through the Factory Class.

1. As the name implies, Factory Class produces the Objects
2. Factory class is a Concrete Class with constructor as private
3. getInstance() method should be public static retrun type is of type IterfaceX
4. It may or may not take input arguments. It depends on the design.





Session Introduction:

- J2EE helps us to develop "Dyanamic Web Application" & there are 2 types of Dynamic web Application
  1. User Dependent Dynamic Web Applications
  2. User Independent Dynamic Web Applications

1. **User Dependent Dynamic Web Application**

- These are the Web Applications where "response Data is specific to a User"
  For ex: Gmail, Facebook, Twitter, Linked-In, etc.
- Login & Logout is "Must" for these web Applications
- Also these web applications MUST know "from which Userr it's getting the request"

2. **User Independent Dynamic Web Applications:**

- These are the web Applications where "Respons Data is independent of the User"
  For Ex: Google Search, Google Maps, Youtube, Grepcode, Any Compay Related Web Applications (www.jspiders.com)
- Login & Logout is "Optional" for these web Applications.

- Also these web applications Need not know “from which User it’s getting the request”

Thumb Rule:

- If we must need to login & Logout to access the web application then, it MUST be a “user Dependent Dynamic We Aplication”.

Why HttpSession (a.k.a Session) Functionality?

- Http is a “Stateless Protocol” i.e. it doesn’t maintain a relationship/state between requests. Each request is un-related to any previous requests.
- Also Http Don’t help web application to uniquely identify the user
- Hence even if user sends sequence of request to web application then it will not able to identify, those are from the same user. Also web application will not able to relate them.
- To address to this problem, Servlet API provides “HttpSession functionality”. With HttpSession we can overcome the above problems.

Thumb Rule:

- Any web application which has Login Page requirement then we MUST make use of HttpSession fuctionality.

HttpSession

- A session in a web application is a time difference between Login & Logout
- "HttpSession" is a functionality provided by Servlet API which helps web application to
  1. Uniquely identify the user
  2. It maintain "State/ Relationship" between requests with the help of "session Attributes"
  3. Provide Authentication for each Request
  4. Tightly couples the different pages of web application.

- To make use of "HttpSession" functionality we must follow 3 steps
  1. Create a Session (Only Once; during Login)
  2. Validate a Session (Many Times; for every request)
  3. Invalidate a Session (Only Once; during Logout).
- HttpSession uses one of the following two mechanisms to handle Session
  1. Cookies (it's default)
  2. URL-Rewriting

<u>HttpSession Related Methods</u>

1. HttpSession HttpServletRequest.getSession()
2. HttpSession

HttpServletRequest.getSession(boolean Create)

<u>Creating a New Session Object Means:</u>

A. Container generates Unique ID

B. Container maintains the Status of this ID as “Active” @ ServerSide
C. Container sends this Unique ID to User in the form of Cookie along with the Response where,
   CooKie Name = “JSESSIONID”
   Cookie Value = “Unique_ID”
D. Container caches this session Object in it’s Cache Memory where, key is Unique_ID
   Value is “Corresponding Session Object”

Note:

Container does All the above steps when we create HttpSession Object [i.e when we invoke getSession() or getSession(true)] & it happens behind the scene.

3. Void HttpSession.invalidate()
➢ Invalidates the current session & garbage collects the associated session object
4. String HttpSession.getId()
➢ This methods returns the unique id. Generated for each session.

5. Void HttpSession.setMaxInactiveInterrval(int timeInterval)
➢ Specifies the valid session time, in seconds
➢ A negative time indicates the session should never timeout
➢ This method helps us to set different Session Time out for different types of users.

6. String HttpServletResponse.encodeURL(String url)
7. String HttpServletResponse.encodeRedirectURL(String url)
➢ Both methods encodes the specified URL by appending session ID to it.

8. setAttribute, getAttribute, removeAttribute

Steps to Handle HttpSession

1. Create the session:
   Whenever user login & after successful authentication create session for the Fritst time.
   HttpSession session = req.getSession(true);
   OR
   HttpSession session = req.getSession();
2. Validate the Session
   Once session is created, for subsequent requests validate the session
   Note:
   Any Servlet/JSP which get the request from Browser, then it should have "Session Validation logic"
   ```
   //get the Current Session Object
   HttpSession session = req.getSession(false);
   If(session == null)
   {
   //Invalid session!!!
   //Generate "Login Page With Error info as Response"
   }else{
   //Valid Session…
   //Generate "Proper Response"
   }
   ```
3. In-Validate the Session:
   A Session get Invalidated in Following Ways
   I. When Application / Server goes down
   II. When user logout of the application

   When user wants to logout of the application, then invoke invalidate() method on the Current Session Object

```
//Get the Current Session Object

HttpSession session = req.getSession(false);

If(session != null)

{

Session.invalidate();

}

//Generate “Login Page with Success Message”

//as a Response
```

III. When user is in-active for configured amount of time
There are two ways to configure session time out

i. In Web.xml
ii. Session.setMaxInactiveIntervalAge()

```
<session-config>

    <!—Time in Mins-- >

    <session-timeout> 10080</session-timeout>

</session-config>
```

iii. In Program

```
HttpSession session = req.getSession(true);
//Time in Seconds, ex: 7 Days
Session.setMaxInactiveInterval(7*24* 60*60);
```

URL Rewriting

- While handling user session “URL Rewriting” comes into picture ONLY if We encode the URL’s with “Session ID”
- If We encode the URL’s, then Container Always first attemp to use the Cookies to get Session ID & fall back to URL Rewriting only if Cookie approach fails.

  Note:
  We cannot make use of URL Rewriting methods in “Static Pages (i.e HTML)”