Friday, 7 April 2017

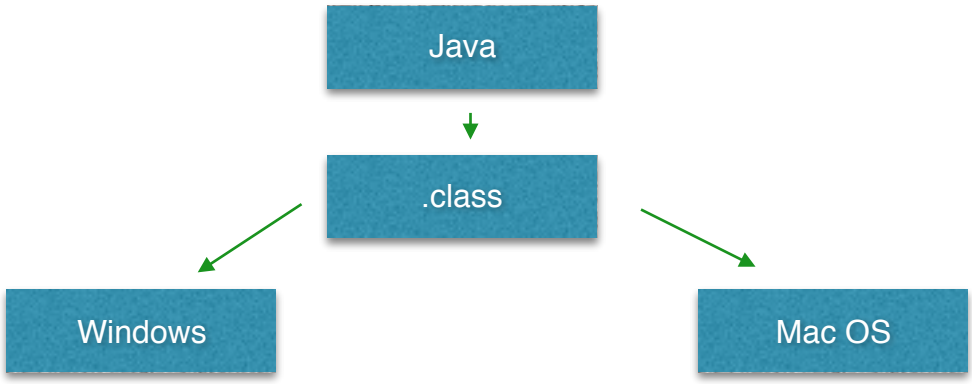
Introduction

Computer :

- It is a combination of hardware + an operating system.
- Computers are represented as shown below :

Application :

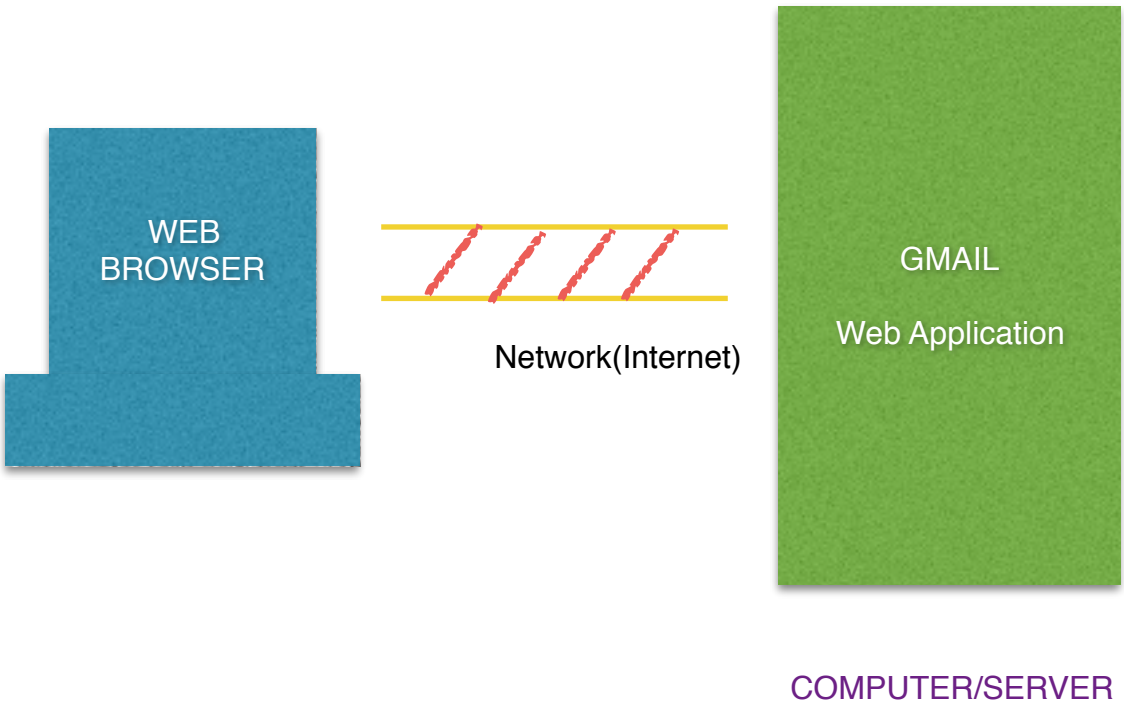
- Every application has its own dedicated functionality . For ex :
  - Adobe Reader can only open pdf files.
  - Media Player can only open .mp3 files.
- Every application has its own know file extensions, For ex :
  - Adobe Reader can only understand .pdf extension.
  - Media Player can only understand .mp3 extension.
- Every application is “Platform Dependent” .
- Java as “Application” is “Platform Dependent” ,however Java as a “Programming Language” is “Platform Independent.
- Java Programming Language is Platform independent because “JVM /JRE” is platform Dependent.



Types of Applications :

- There are 2 types of Applications :
  - Standalone application (unshared)
  - Web Application (shared)
- Standalone Applications are present in our own computer and they are dedicated per user . ex : Adobe Reader , Web Browser , Media Player etc.
- Web applications are not present in our own computer but they are present in some other computer where our own computer and that computer are “Network Connected”

  ex : Gmail , Facebook ,Twitter etc.



- There are two types of Standalone Applications
  - Desktop Applications
  - Mobile Applications

Desktop Applications:

- As the name implies ,they are present in our “Desktop/Computer”.

Mobile Applications:

- As the name implies ,they are present in our “Smart Phone”.

To interact with web application we must make use of

- Network
- Web browser

- Web Browser is a Desktop Application and if we are using browser means we are interacting with Web Application.

- A Network is collection of computers and there are two types of network :
  - Intranet (Private Network)
  - Internet (Public Network)

- Hence Web Applications can be present either in “Intranet” or in “Internet”.

  ex :

  - Gmail Web Application is present in internet.
  - “Jspider Attendance Management System” is present in Intranet.

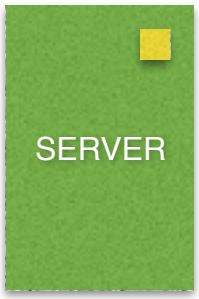
- IP Address / “Computer Name (DNS Name)” Uniquely identifies a particular Computer in a Network.

Server :

- Server is also a computer , but the hardware configuration of this computer is very high when compared to a Desktop Computer.

  ex : Server can have a RAM of 22TB.

- Servers are represented as shown below :



RDBMS Application:

- It`s also an application which helps us to store and maintain GB`s and TB`s of data
- RDBMS Application is represented as shown below :

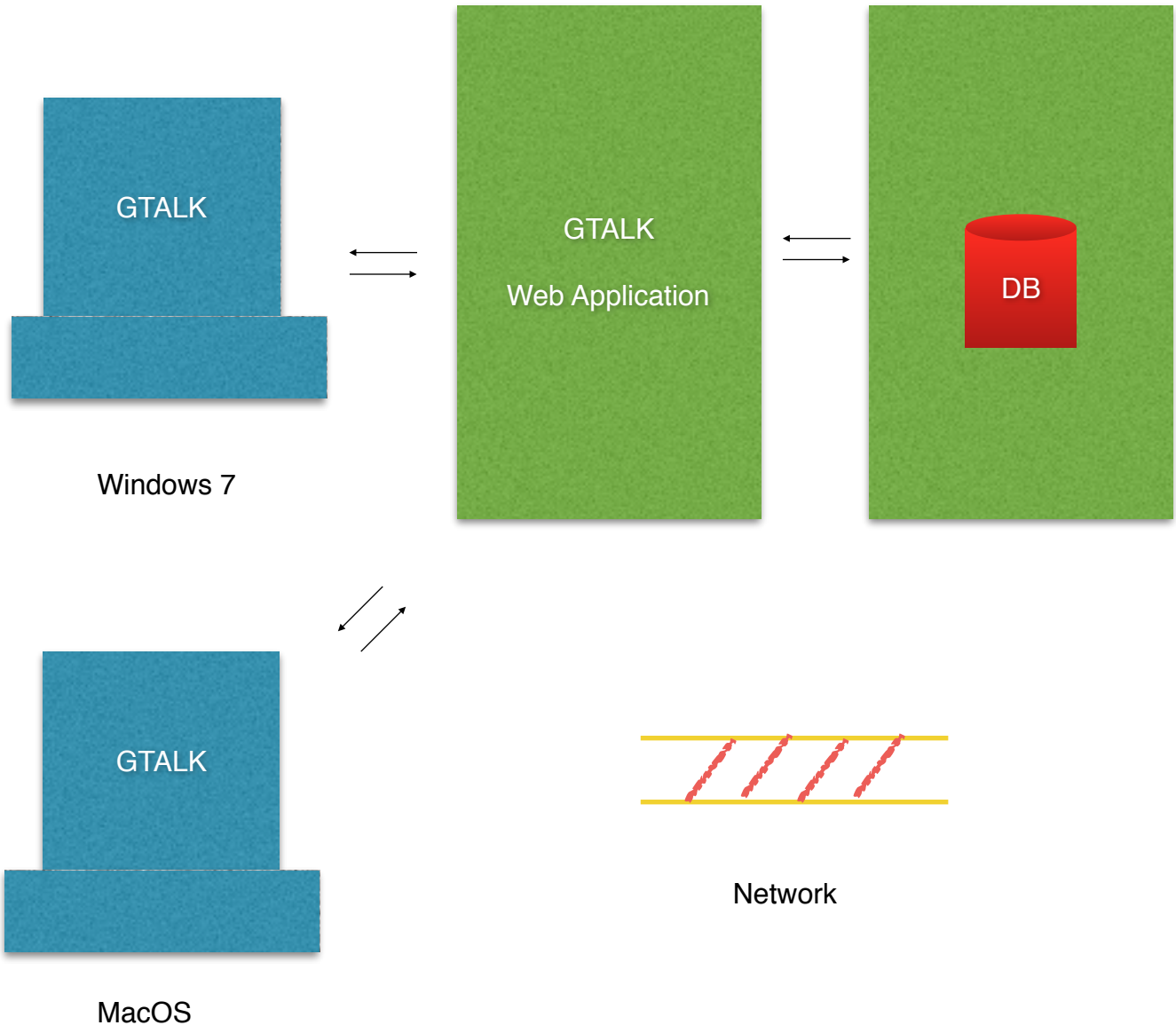
Operating System :

- It acts like an interface between an “Application” and “underlying hardware”.

Java 2 Enterprise Edition / J2EE / JEE / J2EE

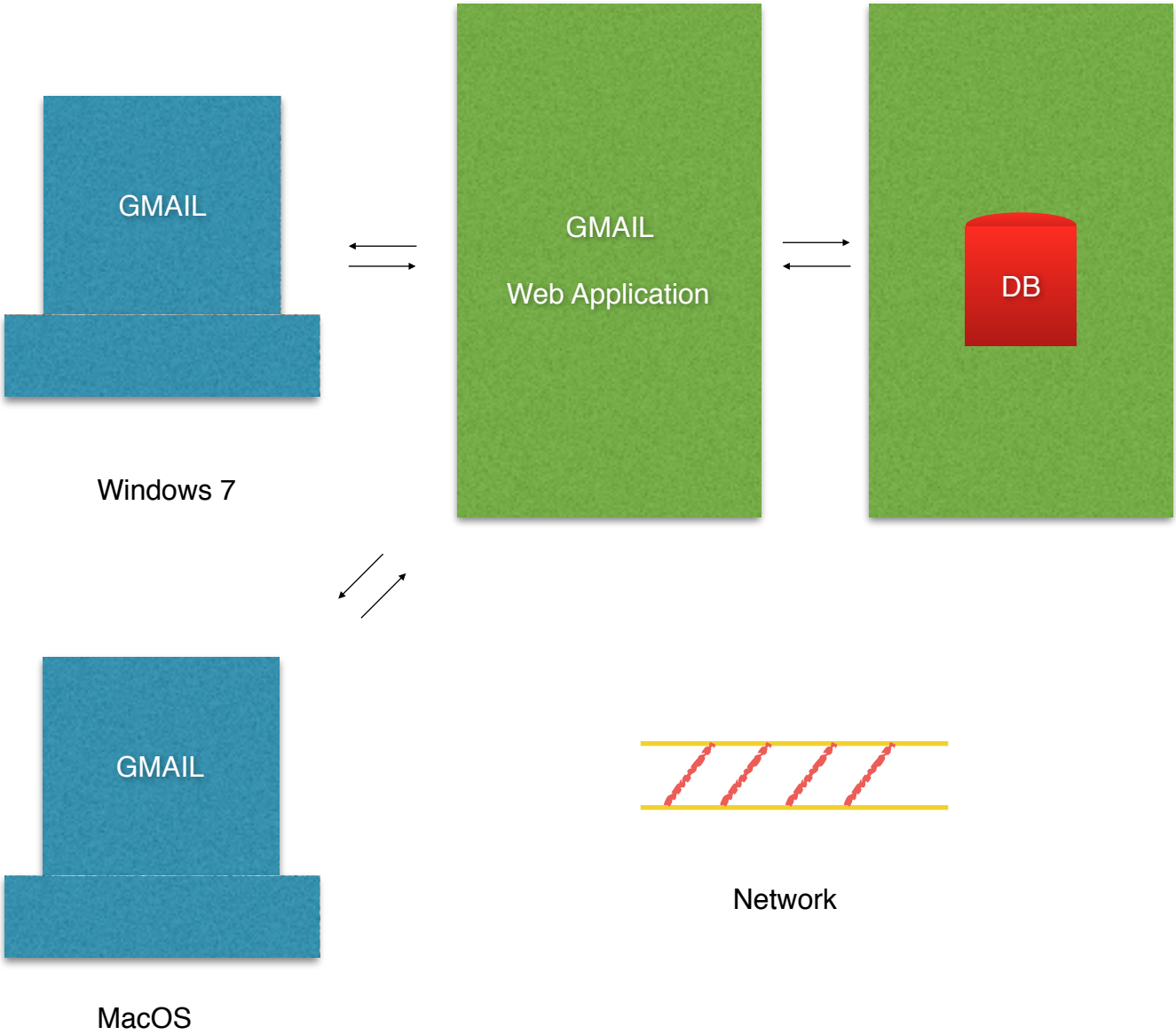
- J2EE is a collection of API`s (Application Programming Interface) , which helps us to develop “Enterprise Web Applications”.
- An API is a “Collection of Packages” with a dedicated “functionality”.
- Major API`s of J2EE are :
  - JDBC
  - Servlet
  - JSP
- JDBC API helps Web Application to interact with RDBMS Application.
- Servlet and JSP API`s helps web application to :
  - get the request from browser.
  - generate the response and
  - gives it back to the browser
- J2EE helps to develop “Web Applications” where as “Android / IOS” helps to develop “Mobile Apps”

Two - Tier Architecture Applications



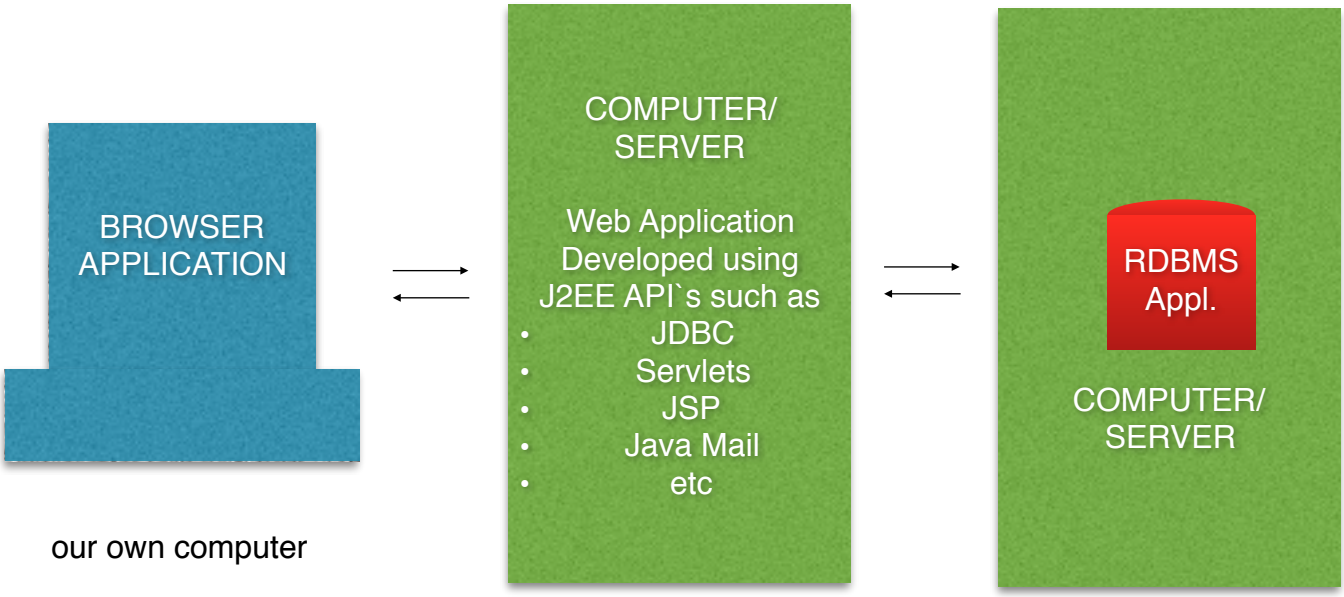
- Download and Install “GTALK” application
- Standalone Application
- Platform Dependent

Three - Tier Architecture Applications



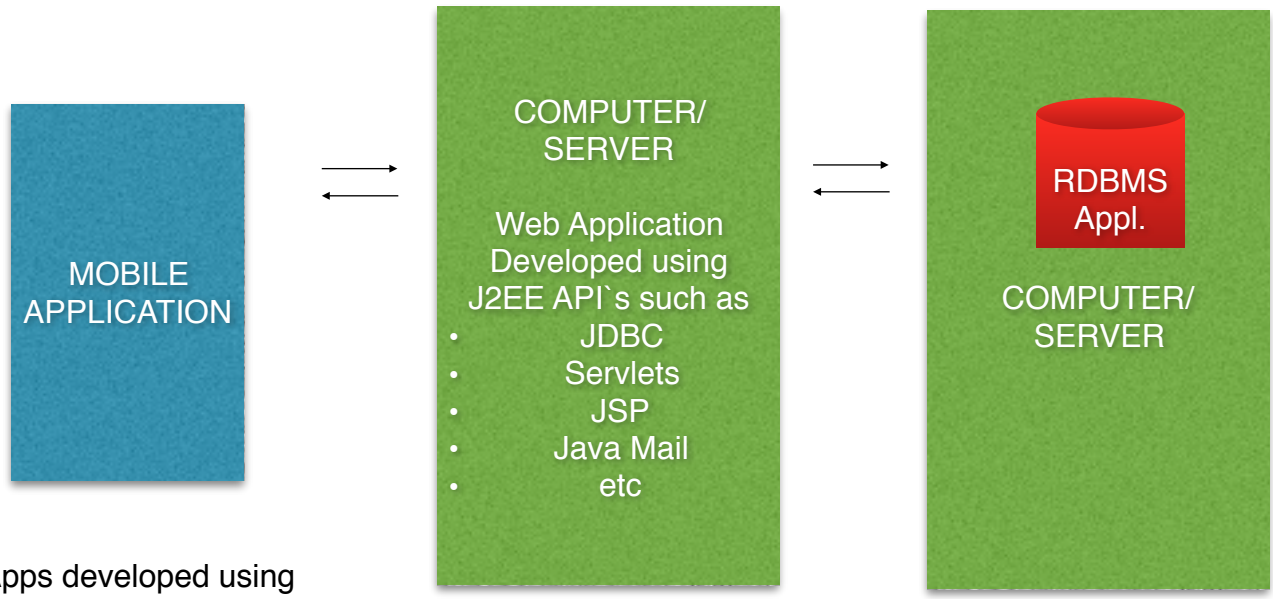
- No Installation Specific to Gmail
- 3-Tier architecture applications are easy to “maintain” as compared to 2-Tier architecture applications.

- ”Mobile Apps” are Two-Tier Applications
- “Web Applications” acts like :
  - 3-Tier architecture application if it is accessed using “Web Browser”
  - 2-Tier architecture application if it is accessed using “Mobile App”



our own computer

CLIENT TIER BUSINESS-LOGIC TIER DATA-TIER



Apps developed using

mobile OS

CLIENT TIER SERVER TIER

JDBC - Java Database Connectivity

one and only API

(DB Independent)

- Java Database Connectivity is an API , as the name implies ,it helps to achieve the connectivity between Java Programs & Database.
- Note : Servlets & JSP`s 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 “One & Only API” that helps to interact only with RDBMS (DB) Application.
- Also JDBC is “DB Independent” i.e using JDBC we can interact with any RDBMS Applications exist in the world.

JDBC Pre-requirements :

- Install an RDBMS Application (MySQL 5.5)
- Create a “Database (Schema)” by name “BECM13”
- Create a table by name “STUDENTS_INFO” in the above database “BECM13”
- Insert some dummy data into the above table

My SQL queries:

```
> create database BECM13;                    // creates database by given name
> use BECM13;                                // to open the particular database
> create table students_info
   (regno int(10) not null ,
    firstname varchar(50) ,
    middlename varchar(50),
    lastname varchar(50),
    primary key(regno));                     // creates table by name students_info
> show databases;                            // shows list of databases present
> show tables;                               // shows table in particular database
> describe table_name;                       //shows table structure
```

Eclipse Shortcuts :

- ctrl + shift + L //shows useful shortcuts
- ctrl + space //autocomplete
- ctrl + F11 //run

Necessary Steps to work with JDBC

- Load the **Driver**
- Get the **DB Connection** via **Driver**
- Issue **SQL Queries** via **Connection**
- **Process the Results** returned by **SQL Queries**
- Close all **JDBC Objects**

Note :

- All the above steps are “**Mandatory**” and “**Interdependent**”
- “java.sql.* ” is the Package representation of JDBC.
- i.e Any Class / Interface belongs to this package means , it`s part of JDBC.

Java Thumb Rules :

- If L.H.S is “Equal” to R.H.S , then L.H.S is always a “Concrete Class”

```
A ref = new A();
```

- If L.H.S is “Not Equal” to R.H.S , then L.H.S can be a “Interface / Concrete class / Abstract Class”

```
B ref = new A();
```

- A class which is declared with “abstract” keyword s called as “Abstract Class”
- A class which is declared without “abstract” keyword s called as “Concrete Class”
- Hence all “final classes” are “Concrete Class”
- Simply having an Interface is of “No Use” & there MUST be “Atleast one Implementation Class”
- Simply having an Abstract Class is of “No Use” & there MUST be “Atleast one SubClass”
- In Java , “Super Class” can either be an “Abstract Class” or “Concrete Class”
- Anything in java which starts with “Upper Case Letter” is either “Class Name” or “Interface Name”
- Anything in java which starts with “Lower Case Letter” & without “Parenthesis” is a “Variable”
- Anything in java which ends with “Parenthesis” is called “Method Name”
- In Java ,anything apart from “Primitive Data Types” are called as “Object References”
  - For ex : int i;

```
boolean isTrue;

Abc ref;
```

Where,

- “i” & “isTrue” are Primitive Data Types
- “ref” is a Object Reference Variable

  where “Abc” can be an **Interface** OR an **Abstract Class** OR **Concrete Class.**

- If a method / constructor input argument is other than primitive data type , then it can be “Interface / Concrete Class / Abstract Class”
  - For ex :

```
public void myMethod(Abc ref)
{
        //Some Logic
}
```

Where ,

- “ref” is a Object Reference Variable
- “Abc” can be

  an **Interface** OR **Abstract Class** OR **Concrete Class**

- If a method return type is other than **primitive data type** , then it can be **Interface** OR **Abstract Class** OR **Concrete Class**
  - For ex :

```
public Xyz myMethod(Abc ref)
{
        //Some Logic
}
```

Where,

- “Xyz” can be

  an **Interface** OR **Abstract Class** OR **Concrete Class**

- There are two types of Interfaces :
  - Marker Interface
  - Non Marker Interface
- If a class implements an interface & we are getting the compilation error means it must be a Non-marker Interface & this interface should have **one or more Abstract methods**
- If a Class implement an interface & we are NOT getting any **compilation error** means it **can be Marker/Non-Marker Interface.**
- If it is a Non-Marker Interface then it should not consist of **ANY Abstract Methods** but it will **consist of only variables.**

Drivers :

- Driver is additional software component required by JDBC to interact with RDBMS application.
- Drivers are provided by DB vendor & they are "DB dependent".
- I.e. Using My-SQL Driver we can ONLY interact with MY-SQL RDBMS Application & Using DB@ Driver we can only interact with DB2 Application.
- "Drivers", as the name implies, acts like a "Bridges" between java application and RDBMS Application.
- DB vendor provider's driver software in the form of a "JAR File".

JAR (**J**ava **A**chieve) file: [**jar**]

- It's a collection of ".class" files + other Necessary resources(text file, XML, Property files, etc.,)
- JAR file helps us to transfer the "JAVA files/ .class files/ 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 files:

1. Right click on the java project, where we want to make use of jar file, select "Build Path" & click on "Add external Achieves".
2. Select the "Jar file" & click on "open".
3. We see JAR file under the "Referenced Libraries".

Program for JDBC Connection

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.
- By referring Driver Manual we can get the “Driver Class” information.

Steps to load the “Driver Class” in to the program :

There are two ways to load the Driver Class :

- By invoking “registerDriver” class by passing an instance of “Driver Class”
  - Syntax :

```
public void DriverManager.registerDriver(java.sql.Driver driverRef) throws SQLException
```

  - Code for MySQL Driver :

```
java.sql.Driver driverRef = new com.mysql.jdbc.Driver();

DriverManager.registerDriver(driverRef);
```

- Second approach to load the Driver Class is with the help of JAVA`s “Class.forName()” Method
- This is the most common approach to register a Driver Class which helps us to pass “Driver Class Name at Runtime”
- But if we create an instance of driver class using new operator , then class name can`t be passed 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 DriverODBC
5. —> Open DataBase Connectivity

D

Type 1 : JDBC - ODBC Bridge

Requires Client Side Installation

Platform Independent

Performance Overhead

Type 2 : Native-API Driver

Type 3 : Network-Protocol Driver

Supports more than 1 DB

Less Performance

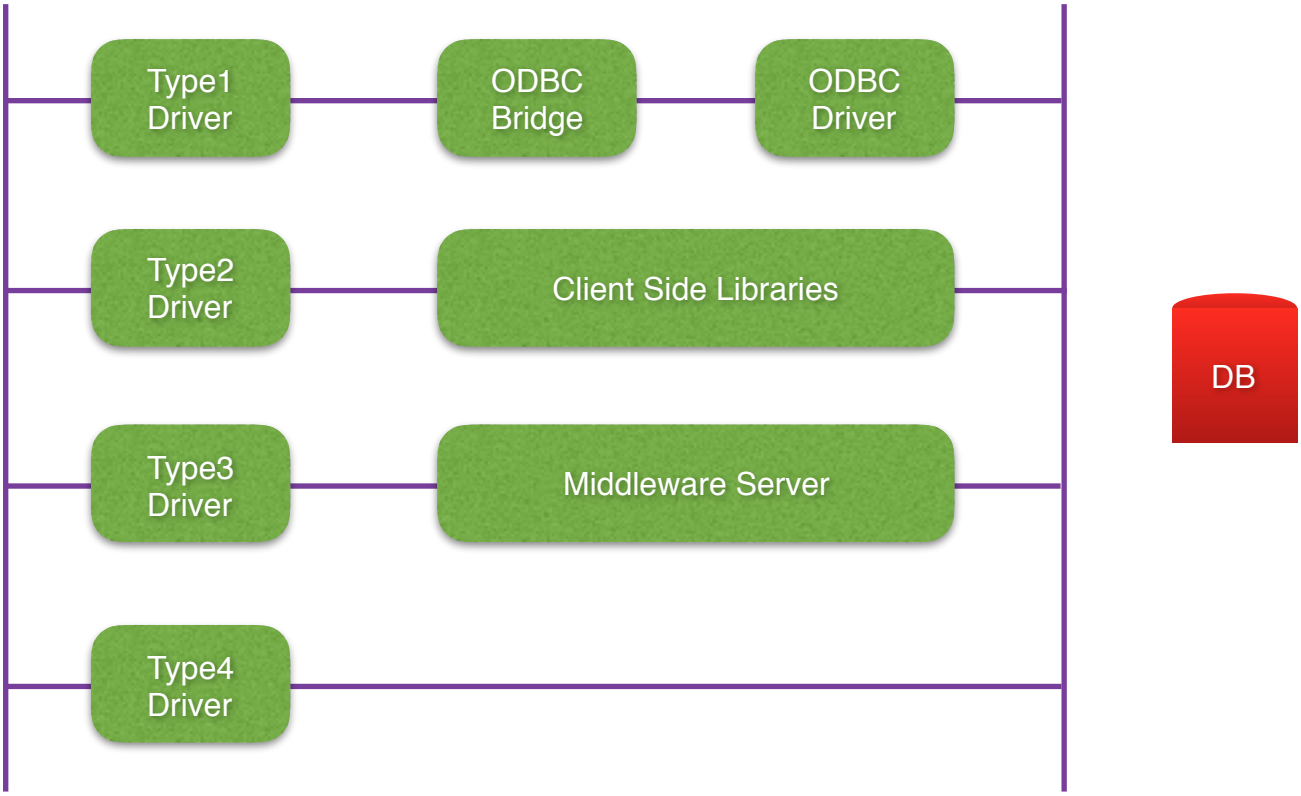
100% pure java drivers

Platform Dependent

Type 4 : Native-Protocol Driver

High Performance

DB Independent



DataBase Uniform Resource Locator (DB URL)

- DB URL , as the name implies ,it uniquely identifies Database OR a RDBMS Application in the Network.
- The structure of DB URL is : <Protocol> : <Subprotocol> : <Subname>
  - Protocol
    - It`s a mandatory Information
    - In case of Java , Protocol is always “jdbc”
  - Subprotocol
    - It`s a mandatory information
    - It identifies the “DB Connectivity mechanism” used to connect to DB
    - This information is provided by DB Vendor & we have to refer the Driver Manual to get this information
    - In case of MySQL , Subprotocol is “mysql” but ,in case of oracle or DB2 it will be different.
  - Subname
    - It`s a mandatory information
    - It Consists of :
      1. Host Name (Computer Name / IP Address)
      2. Port Number (3306 - MySQL)
      3. Database Name /Schema Name (becm13)
      4. User Name & Password (root - root)
    - Arrangement of Surname varies from driver to driver , we have to refer the manual & arrange accordingly.
    - Port Number :
      - It uniquely identifies an application in a Operating System.
      - In case of DB URL ,it uniquely identifies a RDBMS Application

- JDBC URL - Few Examples :
  - Oracle :

    jdbc : oracle : thin : myUser/myPassword@myHost :1521 :myDB
  - MySQL :

    jdbc : mysql : //myHost : 3306/myDB?user=root&password=root
  - MS-SQL Server :

    jdbc : microsoft :sqlserver ://myHost : 1433 ; DatabaseName=becm13; user=root; password=root

---

java.sql.DriverManager

- DriverManager is a "**Concrete Class**" present in JDBC API & as the name implies ,it manages the drivers
- It helps Java Program to establish the connection to DB & for that it requires following two critical information :
  - Driver Class
  - DB URL
- By invoking "registerDriver()" method on DriverManager we can provide "DB URL"
- DriverManager`s getConnection() method helps us to establish the connection to the DB. This method :
  - throws SQLException if it is unable to establish the connection to DB or
  - returns an object of type "**Connection**" if it is able to establish connection to DB
- "java.sql.Connection" is an interface & It`s an "Object representation of Physical DB Connection" that is used by Java Program to communicate with DB.
- Driver Manager consist of only one constructor which is "**private default**" in nature.
- Hence it cannot be **inherited** or **instantiated**. So whatever the methods it exposes to outside world , the should be "**public static**" in nature.

- DriverManager has overloaded versions of getConnection() methods.They are ,

1. Connection getConnection(String dbUrl) throws SQLException

   String dbUrl = “jdbc:mysql://localhost:3306/Becm13?user=j2ee&password=j2ee”;

   Connection con = DriverManager.getConnection(dbUrl);

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

   String dbUrl = “jdbc:mysql://localhost:3306/Becm13”;

   String userNM = “root”;

   String password = “root”;

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

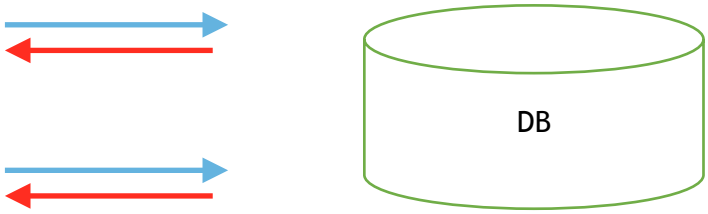
Return type of database

I/P : Other than select

O/P : No. of rows Affected

I/P : Select sql query

O/P : DB result (table)



Results of RDBMS Application

- Whenever we issue “Select SQL Queries” to DB . It returns DB Results
- Whenever we issue “Other than Select Queries” to DB then it returns “No. of rows affected count” in the form of integer
- hence w.r.t results w can group SQL Queries into two groups
  - Select SQL Query
  - Other than Select SQL Query

Static SQL Queries

- SQL Queries
  - “without conditions” OR
  - “with hardcoded condition values” are called “Static SQL Queries”
- Example :
  - select * from ABC;
  - create database DB_NAME;
  - select * from ABC where X = 1;
  - insert into ABC values(1, ‘Praveen’);

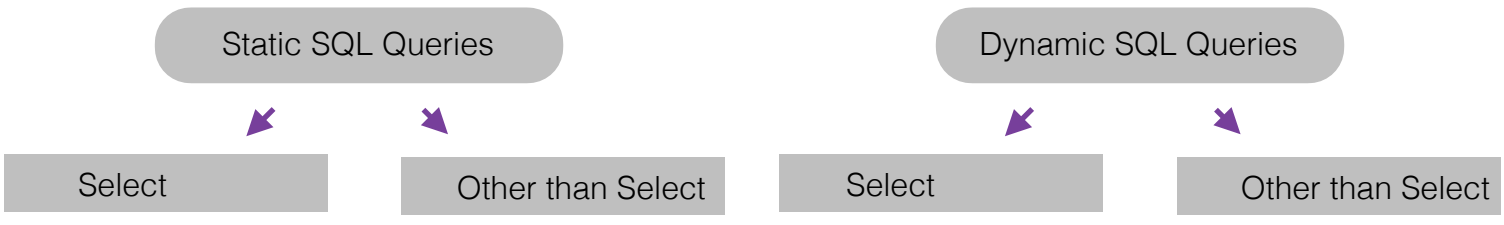
NOTE: ABC = Table name

Dynamic SQL Queries

- SQL Queries which
  - MUST have conditions AND
  - One/More condition get decided at runtime are known as “Dynamic SQL Queries”
- Examples :
  - select * from ABC where X=? and Y=?;
  - select * from ABC where X=1 and Y=?;
  - insert into ABC values (? ,’Praveen’);

NOTE: ABC = Table name

Dynamic SQL Query MUST contain one/more “**?**”



JDBC Statements

- JDBC Statements send SQL Queries to RDBMS and retrieve the data from RDBMS Application
- There are different types of JDBC Statements
  - java.sql.Statement
  - java.sql.PreparedStatement
- Once we create JDBC Statement object (any of the above type) ,then we MUST invoke any of the below method to issue SQL Queries to DB

1 . int executeUpdate() throws SQLException

- This method is used to execute “Other than SELECT” SQL Queries
- This method returns “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.

Statement	int executeUpdate( )
static	other than select
PreparedStatement	ResultSet executeQuery ( )
dynamic	only select

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” Object

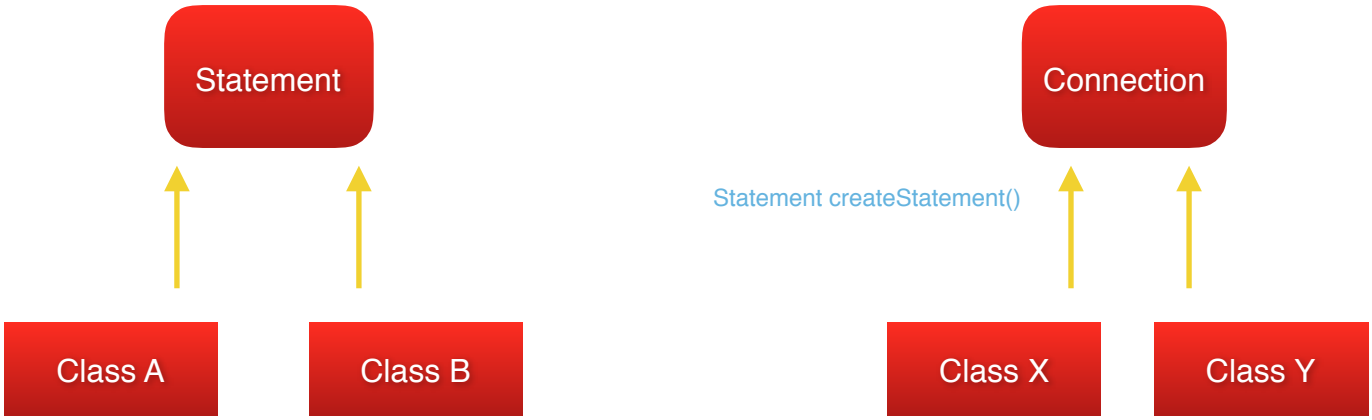
Syntax :

Statement Connection.createStatement() throws SQLException

Example :

Statement stmt = con.createStatement();

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



java.sql.PreparedStatement

- Its 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 = “select * from students_info”+” where region=?”;

PreparedStatement stmt = con.prepareStatement(query);

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

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

Syntax :

void setXXX(Occurrence of ? as int value ,Corresponding Runtime value)

throws SQLException

where XXX = Java Data Type corresponding to DB Column Data Type

Also, “Runtime Value” Data Type SHOULD be same as “XXX Data Type”

- PreparedStatements are also known as “Precompiled Statements” & they help us to achieve “high performance”

Processing the Results returned by SQL Queries :

- Whenever we issue SQL Queries to RDBMS Application via JDBC there are two kinds of results expected out of RDBMS Application
  1. No. of Rows Affected
  2. DB Results
- JDBC returns
  - “No. of Rows Affected Count” as “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 Statement OR PreparedStatement Objects.
- Examples :
  - ResultSet rs = stmt.executeQuery(“SQL Query”);
    - where “stmt” is a Object reference of Statement.

- ResultSet rs = stmt.executeQuery();
  - where “pstmt” is a Object reference of PreparedStatement
- ResultSet consists of N number of Rows with each row containing N number of columns
- No 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 consists 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 follows
  1. Move to desired 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 >);

       OR
     - 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.

Why we need to close JDBC Objects

- JDBC Objects such as
  - Connection
  - Statement ,PreparedStatement and
  - ResultSet

    make use of memory
- In case of Connection Object ,further RDBMS Application resources are consumed
- Also memory consumed by ResultSet Object is comparatively more than other JDBC Objects

- Hence forgetting to close any of the JDBC Objects “will heavily impact Java as well as RDBMS 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 Object invoke “close()” method.
- Syntax :

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

JDBC Summary :

- Commonly used JDBC Objects are :
  1. java.sql.DriverManager
  2. java.sql.Connection
  3. java.sql.Statement
  4. java.sql.PreparedStatement
  5. java.sql.ResultSet
  6. java.sql.SQLException
- 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” & its a “Checked Exception”
- Steps 1 to 4 will in “try block” and step 5 will be in “finally block”
- While making use of JDBC we MUST follow 5 steps and out of 5 ,Only Once
  - We need to load the Driver (Step1)
  - We have to get the DB Connection (Step2)
  - We have to Close JDBC Objects (Step3)
- But ,Step 3 & 4 (i.e Issuing SQL Queries & Processing Results) can happen “N” number of times depending on our need (Min.1 Max.N number)

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

1. Static Web Resources :
   - These Resources are “present at web application” before making the request
   - Content of these resources “do not change” for every request (Static Response)
   - In other words, resources which generates “static response” are called as Static Web Resources.
   - Few Examples :
     - Any songs/movies files downloads.
     - Any software download
     - Any book download
2. Dynamic Web Resources :
   - These Resources are “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 Resource)
   - In other words ,resources which generates “dynamic response” is called as Dynamic Web Resources.
   - Few Examples :
     - Any Banking Web Application Transaction Statement Download (PDF File)
     - Any Post-paid Connection Statement Download (PDF File)
     - Gmail Inbox Page (HTML Page)
     - Google Search Page (HTML Page)

NOTE : Both Static & Dynamic Response can be “HTML” or “Non-HTML” in nature

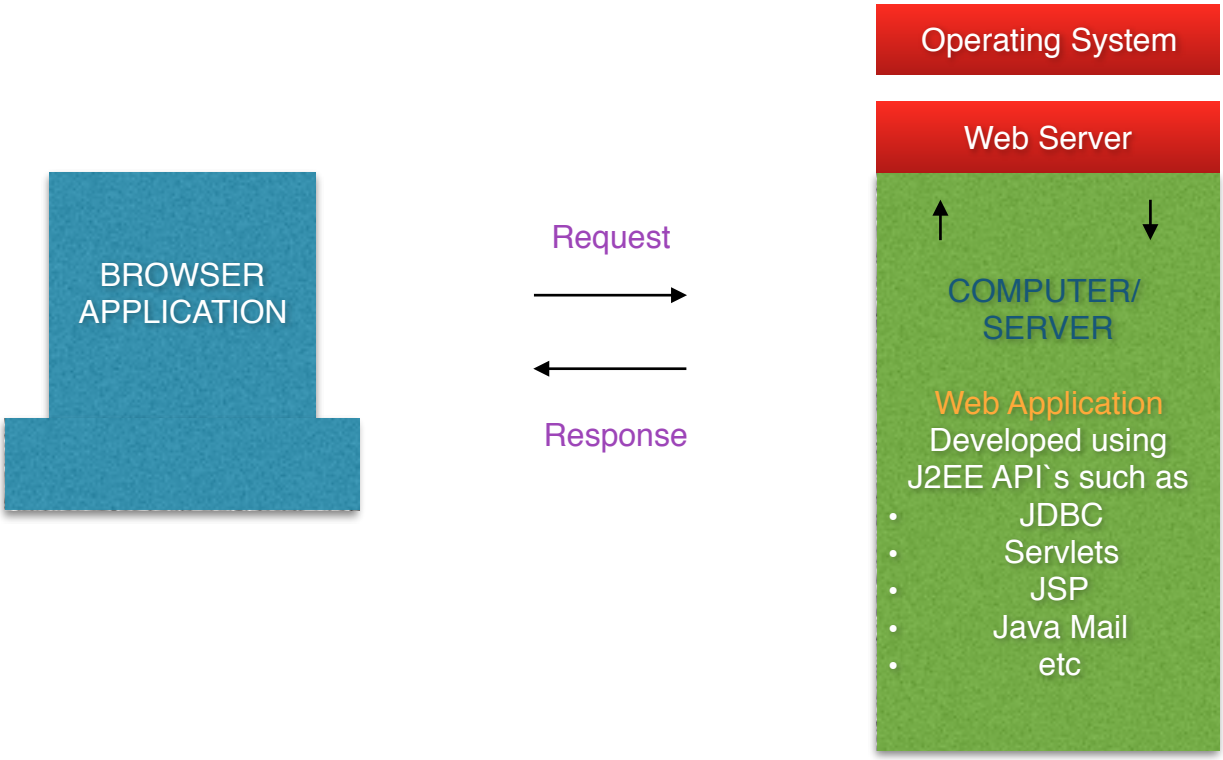
Web Application

- Web Application is an application which is accessed over the network with the help of browser.
- Web Application is a collection of web resources.
- If a Web Application consists of “ONLY static web resources” then it is called as “Static Web Application”
- If a Web Application consists of “one / more dynamic web resources” then it is called as “Dynamic Web Application”
  - Ex : GMAIL ,FACEBOOK etc.
- J2EE / JAVA EE helps us to develop “Dynamic Web Applications” only.
- However HTML helps us to develop “Static Web Applications”

Web Server

- Line any other application (Adobe Reader ,Media Player ,etc) .Web Server is also an application which runs an Operating System.
- Web Server as the name implies “Serves requests to a Web application”
- 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 web server.
- Few examples of Web Servers :
  1. Apache Tomcat
  2. Redhat JBOSS
  3. IBM WebSphere
  4. Grace WebLogic
  5. Grace GlassFish & many more..

- Web Server Image :



Steps to install Apache Tomcat Web Server :

- Download the Apache Tomcat (compressed version) & extract it to some directory.
- Set the following Environment Variables :
  - JAVA_HOME
  - CATALINA_HOME

NOTE : JAVA_HOME should consist of "JAVA Installation" path (jdk)

CATALINA_HOME should consist of "APACHE TOMCAT Installation" path

- Tomcat\bin folder will consist of a file named "startup.bat". Double click it.
- Server will start ,if it throws an exception ,it means it started in "EXCEPTION MODE"
- Likely there will be a problem with environment variables.
- To shutdown :
  - ctrl + c OR Close the console window.

Different ways to interact with Web Applications :

- By Typing an URL in Browser
- By Clicking on the HyperLink
- By Submitting the HTML Form

NOTE :

- Step 1 happens ONLY ONCE
- However , STEP 2 & 3 can happen N number of times

Web URL

- Web URL ,uniquely identifies a particular web resource inside a web application
- In other words , every web resource (Static/Dynamic) must have its unique address in the form of “Web URL”

NOTE :

- In case of Static Web Resources ,URL consist of Resource File Name.

Index.html

==========

```
<html>
<head>
        <title>Index.html</title>
</head>
<body>
        <h1>
                Current Data & Time is :<br>
                <font color = "red">3 - MAY - 2017</font>
        </h1>
</body>
</html>
```

Java Code to Generate Current Date & Time :

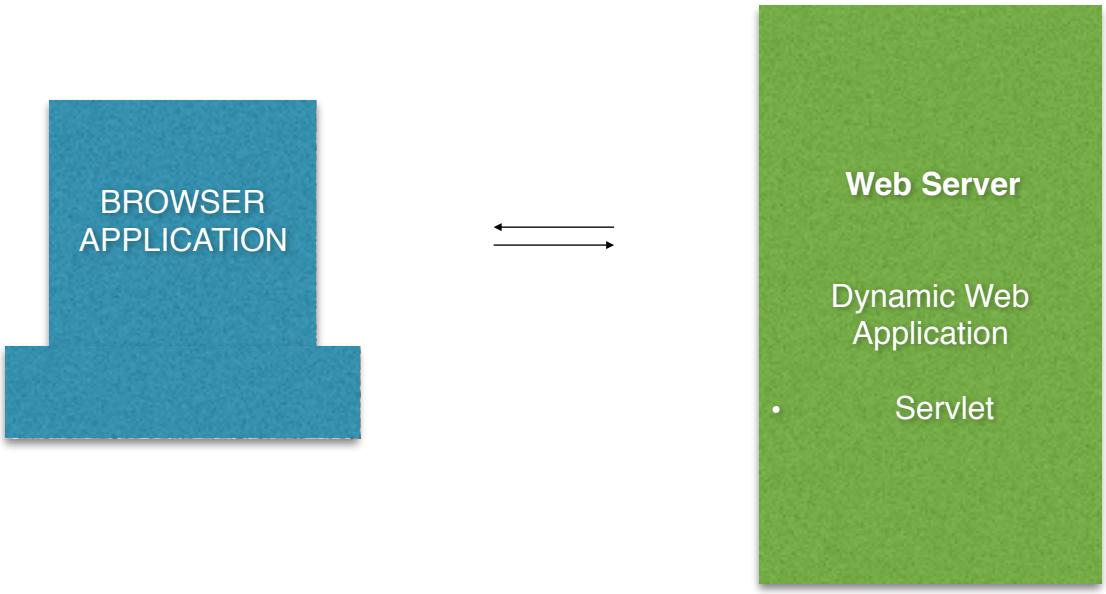
```
import java.util.Date;
public class DateAndTime {

	public static void main(String[] args)
	{
		Date dateRef = new Date();
		String current_date = dateRef.toString();
		System.out.println("Current Data & Time is \n"+current_date);
	}
}
```

Servlets

- Servlets is an API of J2EE ,
  - It accepts web request from Browser via Web Server ,
  - Generates “Dynamic Response” i.e response generated at the time of Request &
  - Gives it back to Browser via Web Server
- Hence Servlets & JSP acts like a “Dynamic Web Resources”
- Since ,J2EE helps us to develop Dynamic Web Application & these web applications should consist of al least ONE Servlet/JSP.
- The Dynamic Response ,may be a “HTML Response” or “Non - HTML Response”
- For Example :
  - Any NetBanking Web application ,Transaction Statement Download (PDF File) is a Non - HTML Response.
  - Google Search Page (HTML Page) is a HTML Response.
- **In the world of Java ,Servlets are the “One and Only API” that accepts web request and generate “Dynamic Response”**
- However HTML helps us to generate “Static Response”

- Since Servlets are like Dynamic Resources & hence Servlets must have its unique address in the form of “Web URL”
- **i.e Even though Servlets are JavaPrograms ,we should not run them like normal Java Programs ,instead we should access via using corresponding Web URL with the help of Web Browser.**



Steps to create “studentsApp” Dynamic Web Application

- Created the “Dynamic Web Project” by name “studentsApp” by selecting web module 2.5 to develop Dynamic Web Application.

NOTE :

- In eclipse we should be in “Java EE” perspective to create Dynamic Web Project
- Dynamic Web Project has 3 following folders :
  - src = “.java” files will be present
  - build = “.class” files will be present
  - WebContent = other than “.java” files will be present

- Created the “index.html” file under the folder “WebContent”.

- Created the servlet under “src” folder by extending HttpServlet.
  - Fixed the compilation error by copying the “servlet.jar” file to “WebContent/WEB-INF/lib” folder.
  - Overridden a method by name doGet( ).
  - javax.servlet.* is the package representation of Servlet API.
  - In case of Dynamic Web Project every JAR file should be kept under “WebContent/WEB-INF/lib” folder.
  - Any class which extends java.servlet.http.HttpServlet is called as Servlet.
- Configured 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
- No need to configure URL for Static Web Resources i.e “index.html”.

- Generate the WAR File from Dynamic Web Project (Build Process)

NOTE :

- To generate WAR File ,right-click on Dynamic Web Project ,select “Export” & click on WAR File.
- In the pop-up chose the destination to keep the WAR File

WAR(WEB ARCHIVE) FILE

- WAR File represents “Dynamic Web Application” & helps to transfer dynamic web application from one location to another.
- It`s a collection of “.class” files
  - other necessary resources
  - & Dependent JAR files

- Transferred the WAR file to Web Server “Web Path” location (Deployment Process)

WEB PATH :

- It`s a Path in WebServer in which Web Applications are present.
- Web path varies from WebServer to WebServer ,we have to refer the manual to get this information.
- In case of Tomcat Web Path is “<Tomcat location>\Webapps”
- Hence in Tomcat ,WAR file should be kept in “Webapps” folder

- 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 a folder by same name inside “Webapps” Folder.

- Accessed the dynamic web application resources using Web Browser by using corresponding web URL`s
- Accessed the HTML Page (Static Web Resource)
  - http://localhost:8080/studentsApp/index.html

NOTE :

- In case of Static Resource URL consists of “Resource File Name”
- HTML generates “Static Response”

- Accessed the Servlet by typing the configured URL present in web.xml (Dynamic Resource)
  - http://localhost:8080/studentsApp/currentDate

NOTE :

- In case of Dynamic Resource URL consists of “Configured URL” present in web.xml
- Servlet generates “Dynamic Response”

MyFirstServlet

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyFirstServlet extends HttpServlet
{
    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp)
            throws ServletException, IOException
    {
        //1. Java Code to Generate Current Date & Time
        Date dateRef = new Date();
        String current_date = dateRef.toString();

        //2. Generate HTML Response
        String htmlRes = "<html>"
                        +"<body>"
                        +"<h1>"
                        +"Curr Date & Time is"
                        +"<br/>"
                        +"<font color=\"red\">"
                        +current_date
                        +"</font>"
                        +"</h1>"
                        +"</body>"
                        +"</html>";
        //3.Send HTML Response to Browser via Web Server
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        out.print(htmlRes);
    }// End  of doGet()
}//End of Class
```

Configuring URL for MyFirstServlet

- In web.xml ,below lines should be present after </welcome-file-list> and before </web-app>
- web.xml

```
    <servlet>
    <servlet-name>myServlet</servlet-name>
    <servlet-class>com.vijay.studentapp.MyFirstServlet</servlet-
class>
  </servlet>

  <servlet-mapping>
    <servlet-name>myServlet</servlet-name>
    <url-pattern>/currentDate</url-pattern>
  </servlet-mapping>
```

NOTE :

- XML Tags are case-sensitive ,for example <Servlet> and <SERVLET> are different compared to <servlet>
- If data present between opening & ending tag contains spaces then ,spaces will be considered
- Hence,
  - <servlet-name> myServlet </servlet-name>
  - is different compared to
  - <servlet-name>myServlet</servlet-name>

Web URL (Web Uniform Resource Locator)

- Every web resources (Static or Dynamic) should have its unique address in the form of Web URL.
- Max number of characters allowed in Web URL is 1024

Web URL Structures

protocol://Domain:Port/Path:QueryString#FragmentId

- **Protocol :**
  - When one application wants to communicate with other (in our case browser and server) ,there needs to be a common language which both applications understands and that language should have set of rules and instructions.
  - This common language is known as **Protocol** where protocol is known as “Set of Rules”.
  - Web Browser and Web Server applications communicate using
    - Hyper Text Transfer Protocol (HTTP)
    - Hyper Text Transfer Protocol Secure (HTTPS)
  - As the name implies most of the time ,HTTP Response contains HTML.
  - In URL it`s an **optional** information and default protocol 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 mandatory information

- **Port :**
  - Port number in Web URL uniquely identifies web server application
  - Default port number for HTTP is 80 and HTTPS is 443
  - In URL it`s an **optional** information
  - When it is not used ,default port number is used depending on the protocol present in Web URL.
  - In Tomcat Web Server ,default port number for HTTP is 8080 and HTTPS is 8443

- **Path :**
  - We know that web application is a collection of web resources (Static/Dynamic) and also web server can consist of one/more web applications.
  - Path in the fun path of the web resources at web application side
  - It consists of ,
  - web application name + (File name in case of Static Resource) OR (Configured URL in case of Dynamic Resource)
  - “Web Application Name” uniquely identifies one web application inside web server.
  - “File Name” uniquely identifies Dynamic web resource inside that web application
  - In URL ,it is an optional information.

Query String :

- Query String is a name & value string pair which passes information ONLY to DYNAMIC Resources such as Servlets & JSP`s.
- 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 represented by an **&**
- Examples :
  - www.google.com/search?q=Vijayraj
  - google.co.in/search?q=ABC&sitesearch=www.youtube.com
  - http ://localhost:8080/studentApp/currentDate?fname=Praveen&lname=s

---

Servlet Code to get Query String Information

- String fnmVal = req.getParameter(“fname”);
- String lnmVal = req.getparameter(“lname”);

Where,

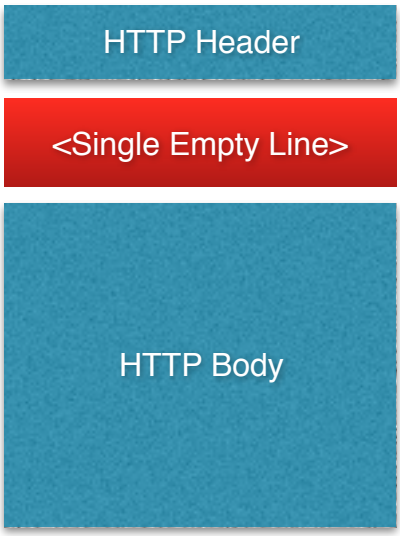
req = Object reference of HttpServletRequest

Fragment ID

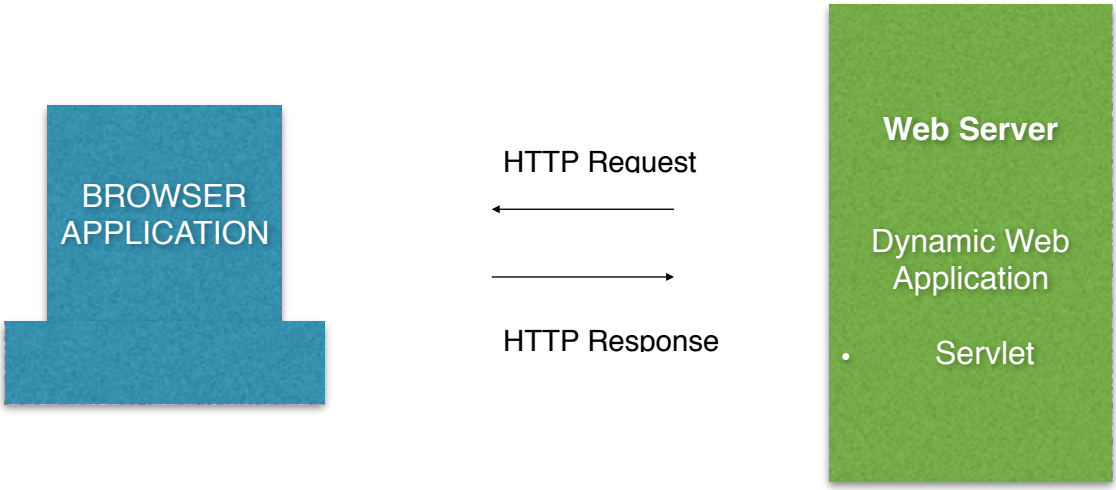
- A Fragment ID or Fragment Identifier ,as the name implies ,it refers to a particular section within a web page.
- In URL ,It`s an optional information & if present ,it begins with a hash (#) character followed by an identifier.
- Example :
  - http://tomcat.apache.org/tomcat-6.0-doc/manager-howto.html#Introduction

HTTP Structure

- Like HTML , HTTP Protocol also has a structure and it consist of Header Part & Body Part.
- Both Header Part & Body Part are separated by a “Single Empty Line”



Key Elements of Request_Response



Key Elements of **HTTP Request** 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.
- 200 : Success / Response is successfully handled
- 404 : File / Resource not found
- 500 : Server encountered an unexpected condition which prevented it from fulfilling the request.
- It`s mandatory information & it will be present in Header of HTTP Response
- Generally WebServer provides “Status Code” info in HTTP Response

Content Type or MIME Type

- Content Type or Multipurpose Internet Mail Extension (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 a Adobe Reader to handle PDF Content
  - Open Media Player to handle media Content etc.

- It`s present in Header of HTTP Response
- The default content type is “text/html”
- Few examples : resp.setContentType(“some type”);
  - text/html
  - application/pdf
  - video/quicktime
  - image/jpeg
  - application/zip etc..

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 web server generates error information & it becomes the “Actual Content”

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 Info) & it will be present in Header of 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 the HTTP Request which depends on HTTP Method present in the request.**

- Size of the Form Data varies from “Form to Form”
- For example :
  - In Case of Gmail Login ,it consists of few kb`s of form data
  - In case of Naukri Profile ,it consists of few mb`s of form data
  - In case of Youtube Video Upload ,it consists of few mb`s to few gb`s of form data

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 Methods & HTTP 1.1 has five new methods. So in total HTTP 1.1 has 8 different methods & every HTTP Request should consist of “ONE OF 8 HTTP METHODS”
- 8 HTTP Methods are :

1. HEAD
2. TRACE
3. PUT
4. DELETE
5. OPTIONS
6. POST
7. GET
8. CONNECT

Code to remember : 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 java.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 (generally we override doGet() and doPost() method)
- Depending on the HTTP Method present in the request ,WebServer invokes the corresponding doXXX(HttpServletRequest req, HTTPServletResponse resp) method.
- In other words ,HTTP method indicates the desired action to be performed on the Dynamic Web Resources . i.e Servlet

NOTE : Since method signature is same apart from Method name ,these methods in general called as doXXX(HSR ,HSR) methods

protected void doXXX(HttpServletRequest req, HTTPServletResponse resp) throws ServletException ,IOException

---

What determines whether the browser sends GET or POST requests

1. Typing a URL in Browser makes request to contain GET method
2. Clicking on a Hyperlink in Browser makes 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 without 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 Password. ex :GMAIL LOGIN
  - Very Large Data like sending mail (gmail compose mail option)
  - OR both of them

  then that form “SHOULD” use method = “post”
- If a form collects “In-Sensitive Data” like search words (ex:google search) then that forms may use method = “post” or method = “get”
- **Hence whenever Servlet gets a request via Submitting the form generally we override doPost() method & for rest of the cases ,we have to override doGet() method.**

Differences between **Get/doGet() & POST/doPost()**

GET	POST
GET is default .	POST is not default.We have to explicitly declare method = “post” in the HTML form
GET requests don't have a “body” or have “empty body”	POST requests have “body”
In case 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
In secure ,because form data will be exposed in url	Secure ,cause data will be present in “body”.hence ,not exposed to outside world
The amount of data sent using Get is restricted because URL can contain Only 1024 characters	There is no restriction on the amount of data send using the POST
We cannot send the files using GET	We can send entire files using POST
We “Can bookmark” GET requests	We “Cannot bookmark” POST requests

Servlet Container

- Servlet Container is a sub-component of web server that helps both web server & server 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 & referring web.xml 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. Once doXXX() method execution is over ,container converts the response object to “Raw HTTP Response” & gives it back to web server
7. Once the response has been given back , Servlet Container garbage collects the request & response objects.
8. In other words **for** ever request ,container creates **new** request & response objects
9. i.e the **Life Span** of these Objects is **Created once request comes** to Servlet and **Destroyed once the response is given back**.

Advantages of Servlet Container

1. **Communication Support** : Container helps both web server & servlets to communicate with each other.
2. **Multi-Threading 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 behaviour 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 Support :** Container takes care of converting JSP into a Servlet.

---

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 info from "**Raw Http 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 :
  - String HttpServletRequest.getMethod( )
    - This method **return** the HTTP Method present in the request as a String Value.
  - StringBuffer HttpServletRequest,getRequestURL( )
    - This method **return** the URL present in the request as a StringBuffer.
  - String ServletRequest.getProtocol( )
    - This method **return** the Protocol present in the request as a String value.
  - String ServletRequest.getParameter(String name)
  - String[ ] ServletRequest.getParameterValues(String name)

NOTE :

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

javax.servlet.http.HttpServletResponse

- "javax.servlet.http.HttpServletResponse" Object ,in short is called as "**Response Object**" ,is an Object representation of **"Raw Http Response"**
- We should make use of this object to send us info as part of **"Raw Http Response"**
- HttpServletResponse is an Interface & it extends another interface by name "javax.servlet.ServletResponse"
- Response Object has Methods which helps us to send information as part of **"Raw Http Response"**
- Methods in **"Response Object"** are :
  - void ServletResponse.setContentType(String contentType)
    - This method helps us to **set content** info to **Response Object**
  - PrintWriter ServletResponse.getWriter( ) throws IOException
  - 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 Object from Response Object by invoking a method by name getWriter( ).
  - PrintWriter has "print( ) /println( )" methods which helps us 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 actual size of the Actual Content there by increases the performance.
- void HttpServletResponse.sendError(int statusCode) throws IOException
- void HttpServletResponse.sendError(int StatusCode ,String errMsg) throws IOException
  - As the name implies ,both these methods helps us to send the Error Response.

eXtensible Markup Language (**XML**) Introduction

- As the name implies it`s an extension of Markup Language & this Language helps to Transfer Data between different Applications
- XML looks similar to HTML but it`s not a HTML

Comparison Between HTML and XML

- HTML helps us to display the Data in the Browser ,XML helps to Transfer/Exchange the data between Applications
- HTML has Pre-Defined Tags ,XML has User-Defined tags.
- HTML tags are “Case-Insensitive” ,XML tags are “Case-Sensitive”
- First Line of HTML is <!DOCTYPE html> (optional) ,XML first line is called as **PROLOG** and it looks like <?xml version = “1.0” encoding=“UTF-8”?>
- HTML is “Not Strictly Typed” Language ,i.e Every opening tag need not have a closing tag and the order in which tags are opened need not be closed in order
- XML is “Strictly Typed” Language.
- File extension of HTML is “.html” ,XML file extension is “.xml”

NOTE :

- Since XML consist of User-defined Tags ,these tags information is defined in an another file by extension “.xsd” (XML Schema Document)
- Hence every application must obey the rules defined in XSD file while constructing the XML file & reading the data from XSD file.

Deployment Descriptor (DD)

- 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**

<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,ten container searches the resources present in this tag in the order they have declared.
- Example :

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file> Static Resource
    <welcome-file>currentDate</welcome-file> Dynamic Resource
 </welcome-file-list>
```

Configuring a URL for 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 the given request
- There must be at least one URL configured for each Servlet. Also Servlet can have more than one URL.
- Below are the different ways to configure the URL for a Servlet
  - Exact Matching
  - Directory Matching
  - Extension/Pattern Matching
- Example :

```
  <servlet>
      <servlet-name>profileServlet</servlet-name>
      <servlet-class>com.vijay.studentapp.StudentProfileServlet</servlet-
class>
  </servlet>

  <servlet-mapping>
      <servlet-name>profileServlet</servlet-name>
      <url-pattern>/studentProfile</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
      <servlet-name>profileServlet</servlet-name>
      <url-pattern>/profile/studentProfile</url-pattern>
  </servlet-mapping>
```

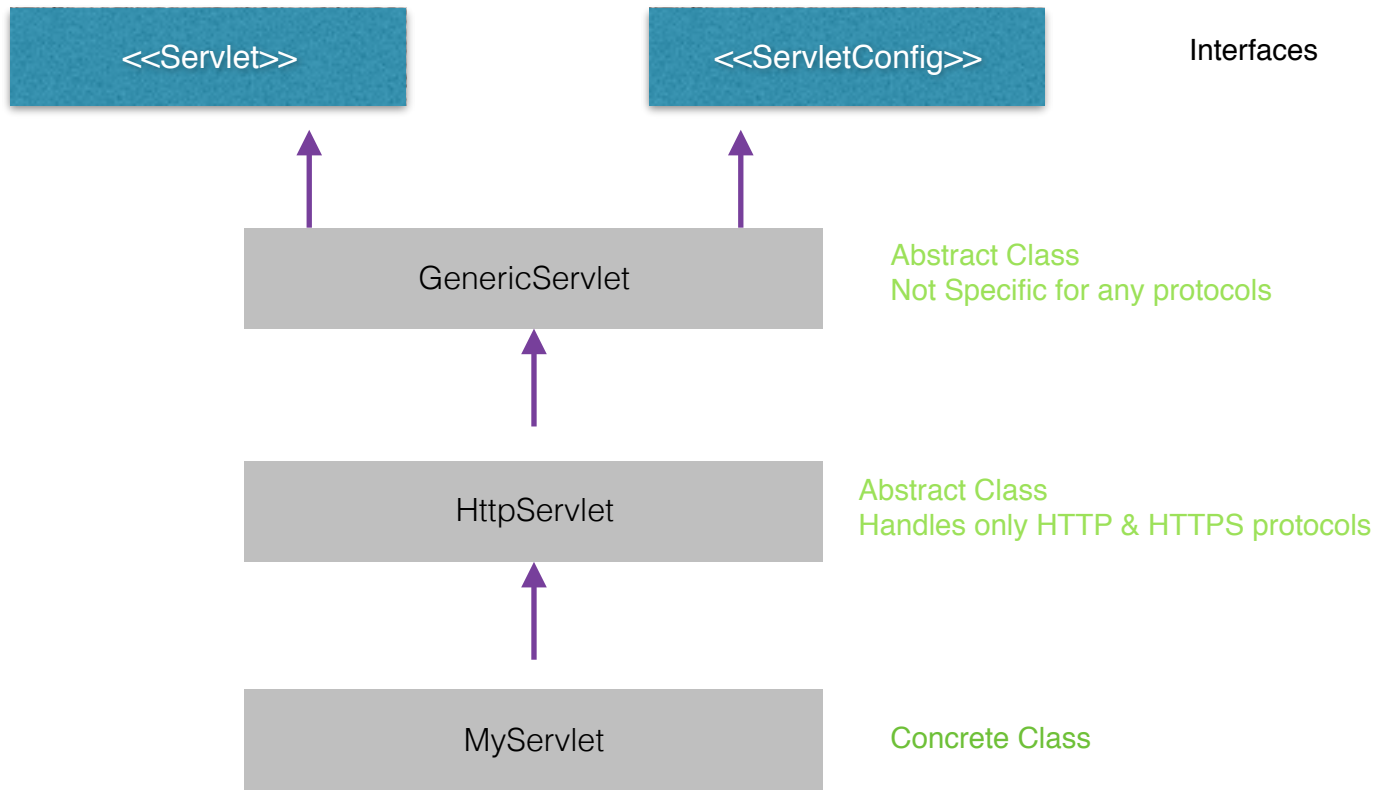
```
<servlet-mapping>
    <servlet-name>profileServlet</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

- Order of preferences
  - Exact Matching
  - Directory Matching
  - Extension/Pattern Matching

Note : We can also configure different URL`s for a same Servlet as shown below :

```
<servlet-mapping>
    <servlet-name>myServlet</servlet-name>
    <url-pattern>/currentDate</url-pattern>
    <url-pattern>/currentDate1</url-pattern>
</servlet-mapping>
```

Servlet Hierarchy



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 void service(ServletRequest req, ServletResponse resp) throws ServletException ,IOException

- Hence whenever we create a servlet by extending GenericServlet we MUST provide implementation for **service( )** method.

- Example :

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class GenericServletExample extends
GenericServlet
{
     @Override
     public void service(ServletRequest req,
ServletResponse resp)
               throws ServletException, IOException
     {
          resp.setContentType("text/html");
          PrintWriter out = resp.getWriter();
          out.println("Inside Generic Servlet");
     }//end of service
}//end of class
```

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**.
- Hence this class MUST provide an Implementation for GenericServlet abstract method. i.e service(SR ,SR)
- 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**” i.e exception message.
  - 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 method return “**Method is not supported by the Servlet API**” error response.
  - A subclass of HttpServlet can override any of the below service( ) method
    - public void service(ServletRequest req ,ServletResponse res) throws ServletException ,IOException
    - public void service(HttpServletRequest req ,HttpServletResponse res) throws ServletException ,IOException
    - public 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/methods.
- All the 7 doXXX(HSR ,HSR) methods in HttpServlet has the logic of Generating “**405 Error Response**”

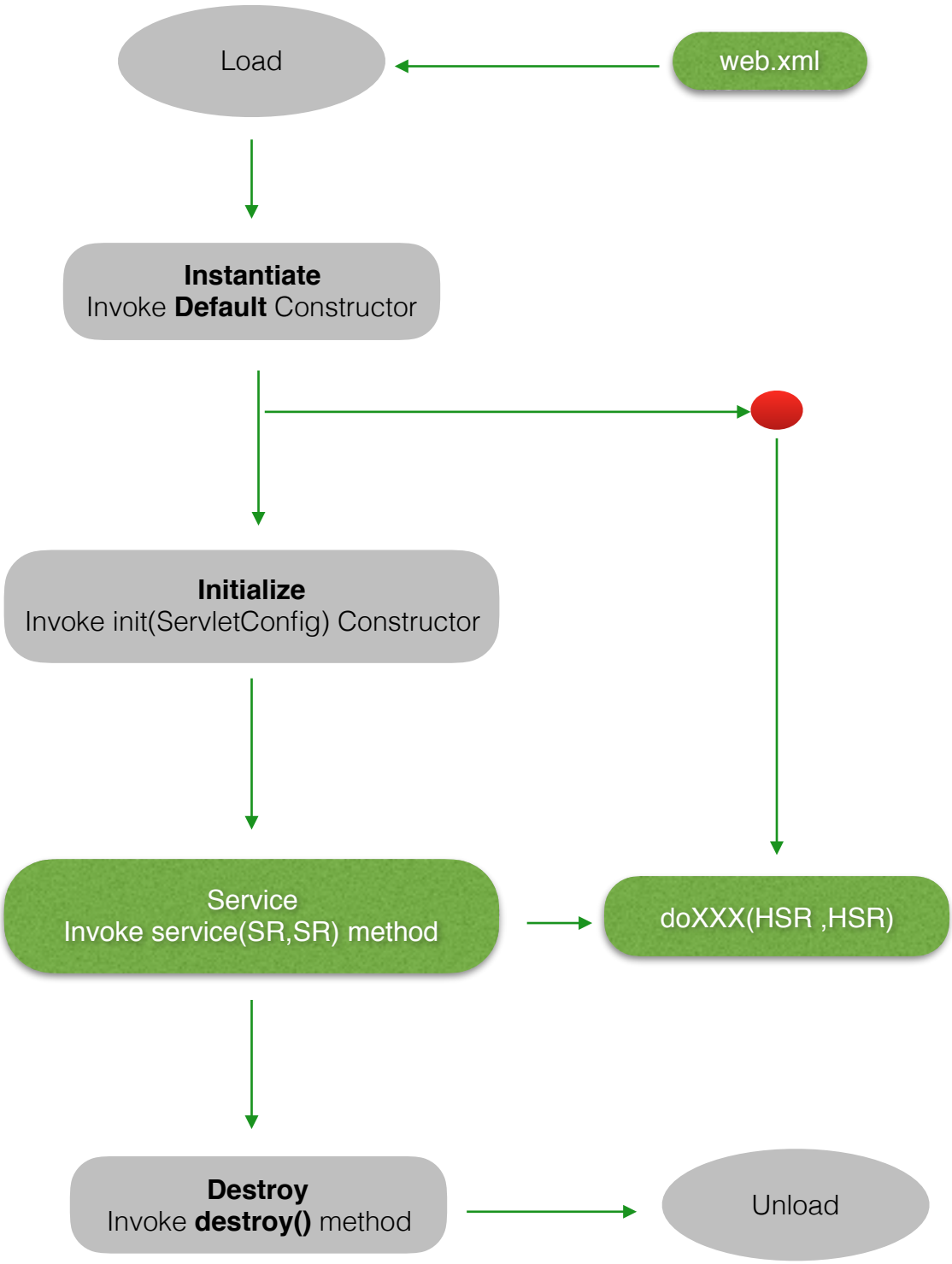
- If we won`t override doXXX( ) methods ,then default implementation from HttpServlet is invoked which in turn return "**405 Error Response**"

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.

GenericServlet	HttpServlet
Protocol Independent	Protocol Dependent .Supports only HTTP & HTTPS protocols
Abstract Class : Because service method is declared abstract	Abstract Class : But no methods are declared as abstract
If we extend Generic Servlet then we must provide implementation for service() method	If we extend HttpServlet ,There is NO restriction on overriding any version of service method but generally we override one or more doXXX() methods.
Generic 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.

Life Cycle of a Servlet



Servlet Lifecycle

- Lifecycle of a Servlet is controlled by
- **Instantiation Phase**
  - Whenever request comes to a container ,by looking at the URL & 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 instance of the Servlet by invoking “Public Default Constructor ONLY”
- **Initialisation Phase**
  - Version 1 :
    - public void init(ServletConfig config) throws ServletException

```
{
super.init(config);
//Initialization code
}
```

  - Version 2:
    - public void init() throws ServletException

```
{
super.init(config);
//Initialization code
}
```

  - 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
  - This method is called ONLY ONCE in Servlet Lifecycle

- We may/maynot override this method. If we don`t override then default implementation present in GenericServlet is invoked.
- The first line of the Version 1 init method should be “super.init(config)”
- During Initialisation , Servlet has access to “javax.servlet.ServletContext” object
- Once Instantiation & Initialisation is successful container caches the Servlet Instance

---

Can we use Constructor for Initialisation ?

- We can make use of constructor for initialisation only if initialisation code is independent of ServletContext Object.
- If Initialisation code is Dependent on ServletContext Object then “We left with no choice of using init method for initialisation purpose”.
- Also, in case of constructor ,we must make use of “public Default Constructor”
- Hence, we generally make use of init method for initialisation purpose (dependent / independent of ServletContext/ServletConfig objects) without touching constructor.

- Service Phase

- public void service(ServletRequest req ,ServletResponse res) throws ServletException ,IOException
- After Instantiation and Initialisation ,Container creates the Request and Response Objects ,invokes service(ServletRequest,ServerResponse) method by passing these objects.
- This method is called “for every request” i.e ,one/more times in “Servlet Lifecycle”
- Inside service(SR ,SR) method we write any Java Code which needs to be executed for every request.
- If a Servlet is a sub-class of GenericServlet then we must override this method.
- If a Servlet is a sub-class of HttpServlet then we should not override this method and our job is to override one/more doXXX() methods.

- **Destruction Phase**
- public void destroy()

```
{
        //clean up code goes here
}
```

- Whenever Container wants to remove the cached Servlet Instance from its memory then it invokes destroy() methods before removing.
- destroy() method gives us a choice to perform any clean up activity such as closing a file etc.
- This method is called only once in Servlet Lifecycle
- We may or may not override this method if we don't override then ,Default implementation present it 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( )

Important Points :

- **Any class which extends any one of the below class is called as a "Servlet"**
  - **javax.servlet.http.HttpServlet**
  - **javax.servlet.GenericServlet**
- **In other words Servlet MUST be an object of type java.servlet.Servlet interface.**
- **If a class extends either HttpServlet or GenericServlet , sub class of that class is also 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 "Installation Phase"**
- **Servlets MUST have public default Constructor OR combination of any other constructor along with public default constructor.**

- **We can also create a Servlet by implementing "java.servlet.Servlet" interface however this approach is not so common. In this case Servlet becomes "Protocol-Independent" in nature.**

- **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.**

- **Servlets can have :**
  - **Local Variables (Static /Non-Static)**
  - **Block of code (Static /Non-Static)**
  - **Inner Classes**
  - **It`s own methods (Static /Non-Static but non-abstract)**

Note :
- If we declare abstract method/methods in servlet then we are forced to declare that servlet as "Abstract Class" & we MUST sub-class it.
- Otherwise it fails at runtime during instantiation.

- **We can have "main() method" in the servlet but its of NO USE. Servlets are directly under the control of Container & Container will not execute main method in any phases of Servlet Lifecycle.**

- **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 environment"**

Single Threaded Servlets

- 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 synchronising Service Method

- SingleThreadModel is a Marker Interface which ensures 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

Ex : public class GenericServletExample extends GenericServlet/HttpServlet implements SingleThreadModel

```
{
        {
                //Logic
        }
}//end of class
```

Note : List of Marker Interfaces

Java/JDK :

1. **java.io.Serializable**
2. **java.lang.Cloneable**
3. **java.util.RandomAccess**
4. **java.util.EventListener**
5. **java.rmi.Remote**

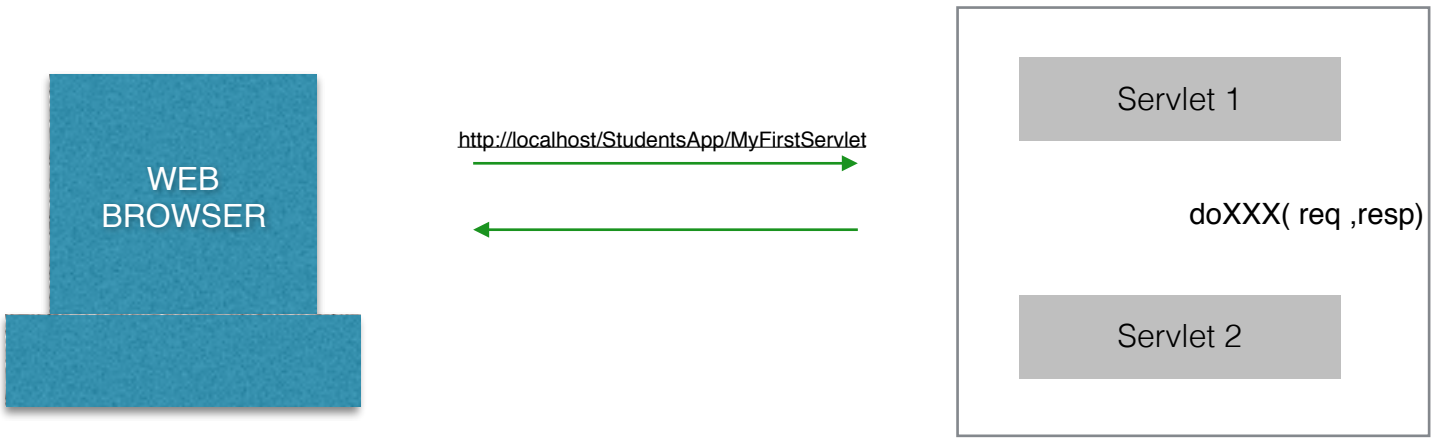
Servlet API : javax.servlet.SingleThreadModel

ServletContext	ServletConfig
ServletContext is an Interface and “an Object of ServletContext” is use by container to pass information to ALL the servlets which are part of an application	ServletConfig is an Interface and “an Object of ServletConfig” used by a container to pass information to a particular Servlet
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 “Destruction Phase”
So there will be “ONLY ONE” Instance of ServletContext Object exists per web application	There will be ONE Instance of ServletConfig per Servlet.
Singleton	Non - Singleton
ServletContext object is obtained by calling “getServletContext()” method we inherit from GenericServlet ServletContext context = getServletContext();	ServletConfig object is obtained by calling “getServletConfig()” method we inherit from GenericServlet ServletConfig config = getServletConfig();

In web.xml ,context parameters are declared under <context-param> tag [One or more]	In web.xml ,config parameters are declared under <init-param> tag [One or more] which is sub tag of <servlet> tag
ServletContext Object “does not” holds the object reference of ServletConfig ServletContext context = config.getServletContext():	ServletConfig Object holds the object reference of ServletContext ServletConfig config = getServletConfig() ;

NOTE :

- Both ServletContext and ServletConfig objects has getInitParameter() method which helps us to get parameter value information from web.xml
  - Syntax :
    - String getInitParameter (String paramName)
- We can ONLY get ServletContext and ServletConfig parameters at Run Time but “we cannot set them”.



1. Using Browser ,user makes a request to a servlet
2. Servlet internally forwards that request to another Internal Resources (Static / Dynamic) by passing Request & Response Objects
3. This internal resource handles that request & gives back the response

- Browser displays the response .In this case Browser will not have any clue what went behind the scene (i.e forward happens at server side)
- Also ,URL in the browser doesn`t change
- To forward the request call forward() on RequestDispatcher object.

Syntax :

void javax.servlet.RequestDispatcher.forward(ServletRequest req, ServletResponse resp) throws ServletException, IOException

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

Syntax:

RequestDispatcher javax.servlet.ServletRequest.getRequestDispatcher(String url)

Example :

RequestDispatcher rd = null;

String url = null;

url = “currentDate”;

```
rd = req.getRequestDispatcher("url");
```

NOTE :

- **We cannot forward the request to External Resources**

For Example : If String url = “www.gmail.com”

REDIRECT	FORWARD
Redirect happens @ “Browser side”	Forward happens @ “Server side”
URL in the browser changes	URL does not change
We can Redirect the request “both Internal & External Web Resources”	We can Forward request “ONLY to internal Web Resources”
Redirect Contains “More than One” request & response cycle	Forward Contains “Only One” request & response cycle
In case of Redirect ,”More than ONE set of Request & Response Objects” get created.	In case of Forward,”ONLY ONE set of Request & Response Objects” get created.
Slower in Operation	Faster in Operation
Redirect makes request to contain GET method & hence it ALWAYS invokes doGet() method at destination	When we Forward the request ,it will invoke the corresponding doXXX() method at Destination.
Redirect happens on “Response Object”	Forward happens on “Request Object”

NOTE :

1. **In both the cases (Redirect / Forward) ,if the corresponding override version of doXXX() method doesn`t exist that invokes default doXXX() method.**
2. **If URL in Browser Changes means it MUST be a Redirect (escpecially Domain Name)**
3. **For Internal URL either we can make use of Redirect OR Forward but "Forward" is preferred**
4. **For External URL we left with no choice we MUST make use of Redirect**
5. **In other words , If one web application wants to communicate with another "via Web Browser without any user intervention",then Redirect is the ONLY way. In this case both the web applications can transfer the data using "Query String"**
6. **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 :
    - public abstract void forward (ServletRequest req , ServletResponse res) throws ServletException , IOException
    - public abstract void include (ServletRequest req , ServletResponse res) throws ServletException , IOException
  - An object of RequestDispatcher helps us to perform either Forward / Include Operation
  - forward() helps to forward 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 overriden version of doXXX() method" in that Servlet
  - If corresponding overriden version of doXXX() methods does not exists ,then Container DOEST NOT give any Compilation Error /Run-time Exception.In this case it just ignore the include statement.

- Ex : RequestDispatcher rd = null;

  PrintWriter out = resp.getWriter();

  out.print(“1”);

```
rd = req.getRequestDispatcher(“index.html”)
rd.include(req,resp);
```

  out.print(“2”);

NOTE :

- We cannot include the response of “External Resources” into the Servlet
- Unlike Redirect & Forward ,we can include more than one Resource/URL at a time.
- Include helps us to reuse the Internal Resources with that its “easy to maintain Web Application”.

Cookie Related Methods

- public Cookie(String name ,String value)
  - Its 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 and
    - should not begin with 4
  - Cookie Name “cannot be changed” after creation
  - Value can be anything & it “can be changed” after creation.
- void HttpServletResponse.addCookie(Cookie cookieObj)
  - This method add the specified Cookie to the Response
  - This method can be called “Multiple Times” to set more than one Cookie to the response
- Cookie[ ] HttpServletResponse.getCookies()
  - This method returns
    - an **Array** containing all the cookie objects OR
    - returns “NULL” if request **does not** have cookies
- String Cookie.getName()
  - This method returns the **Name** of the Cookie
- String Cookie.getValue()
  - This method returns **Value** of the Cookie
- void Cookie.setMaxAge(int expiry)
  - set`s max age of the Cookie in “Seconds”
  - +ve Value makes Cookie “Persistent”
  - Any -ve Value makes Cookie “Non - Persistent”

- 0 makes Cookie (Persistent / Non-Persistent) to be “Deleted Immediately” From Browser.
- Examples :
  - **Create Cookie :**
    - //Non - Persistent Cookie
    - Cookie myLocationCookie = new Cookie(“myLocation”,”Bangalore”);
    - //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”);
  - **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 Cookie info(name & value) in Browser
    - out.print(“Cookie Name :”+rcvdCookie.getName());
    - out.print(“Cookie Value :”+rcvdCookie.getValue());
    - }
    - }//end of if-else

- **Delete 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){
• String name = rcvdCookie.getName();
• //Delete ONLY ‘myLocation’ Cookkie
• if(name.equals(“myLocation”))
• {
• rcvdCookie.setMaxAge(0);
• resp.addCookie(rcvdCookie);
• out.print(“Deleted ‘myLocation’ Cookie”);
• break;
• }
• }//end of for
• }// end of if-else
```

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 :
  - Context Attributes (Application Scope)
  - Request Attributes (Request Scope)
  - Session Attributes (Session Scope)
- These 3 Attributes can be get/set by using following objects respectively
  - javax.servlet.ServletContext
  - javax.servlet.ServletRequest
  - javax.servlet.http.HttpSession
- Following methods are present in above Three Objects , which can be used to “get, set or remove” attributes
- 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 new object
- public void getAttribute (String name)
  - 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.
- public void removeAttribute (String name)
  - This method removes an attribute with the given name
  - After removal ,subsequence calls to getAttribute() with the same name returns NULL.

NOTE : 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. **Attribute**

- Apart from Attributes, rest are name = value pair where both name & value are “String”
- Where as Attributes are name = value pair where name is “String” and value is “Object”
- Hence if one Servlet wants to pass information to an another in the form of “Java Object” then “Attributes” are the One & Only way.

NO.	ServletContext	ServletConfig	HttpSession
1.Instance	One per Application	One per Request & Response Cycle	One per user or LOGIN/ LOGOUT
2.Lifespan	Created during Server Startup and Destroyed during Server Shutdown	Created when Request Comes and Destroyed when Response is given	Created when user login and destroyed when user logs out
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)		