Application

Application is a set of programs which is used to provide solutions for day-to-day life problems

E.g.: To perform the calculations developers introduces calculator

Programs: are set of instructions by which developers perform different functionalities or tasks in an application

We have 4 types of applications:

- **Stand-Alone Application**
- **Web Application**
- **Mobile Application**
- **Distributed Application**

Stand-alone Application:

- It is also known as Independent/Desktop application
- All these standalone applications must and should be downloaded and installed on client machine like laptop \ Desktop
- To run this application, users require OS like windows, Linux, Unix, mac, etc.
- E.g.: Calculator, SQL+, Browser.
    - To develop Standalone applications programmers, use J2SE(Java 2 Standard edition).i.e. Core java.

Web Application

- The application which requires internet and browser is referred as Web Application.
- Web application does not require installation in a client machine.
- All the web applications are present in web servers.
- To develop web application, Programmers are using J2EE (Java 2 Enterprise Edition). i.e. Advanced Java.
- e.g.: Amazon/Flipkart, Netflix, YouTube.

Note:

- Browser is a standalone application which is used to access web applications with the help of internet. E.g.: Chrome, Firefox.
- J2EE is an extension of J2SE which is developed web applications, microservice, Rest API's.
- J2EE consists of following concepts:
    - Servlets
    - Java Server Pages
    - Hibernate framework
    - Spring Framework

Servers:

Server is a machine/ Computer which is acts like a mediator between programmer & users.

There are multiple types of servers:

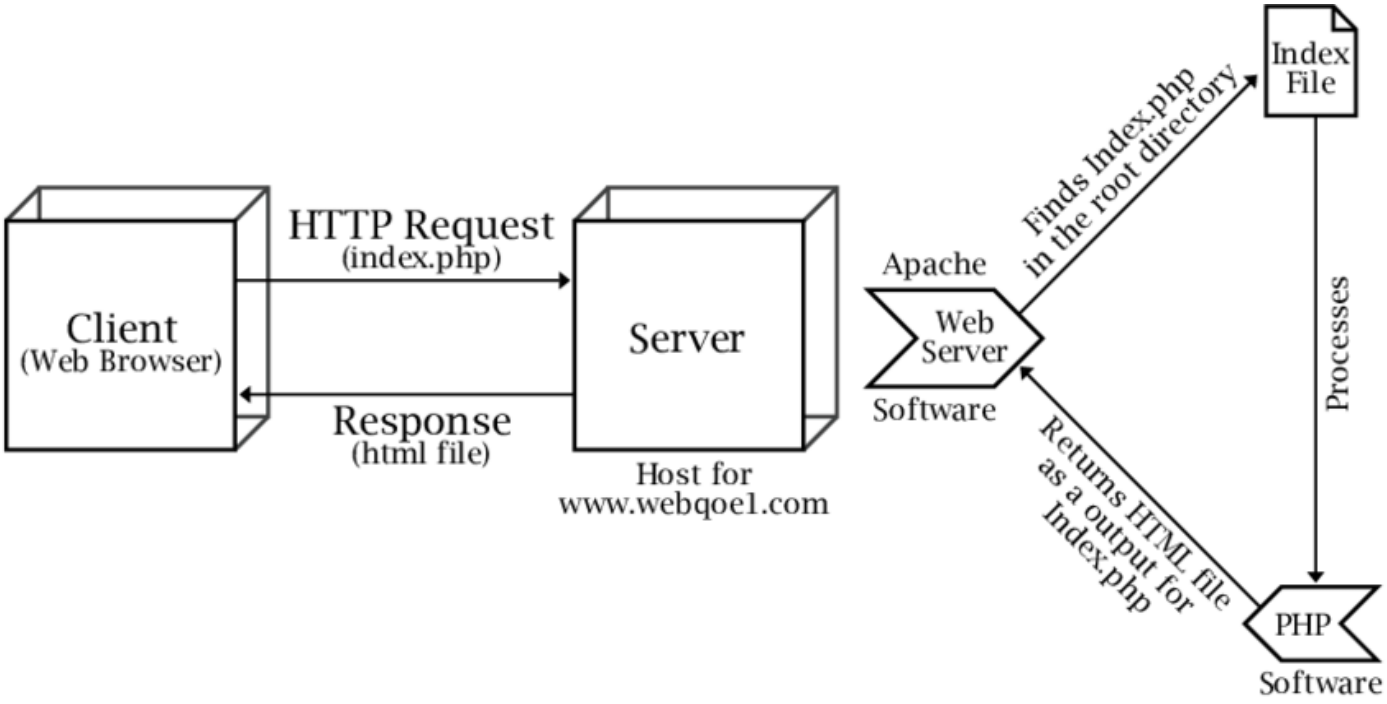
Web Server: used to store web application

Database Server: Used to store data.

Application Server: used to store microservices & Rest API.

Mail Server: used to send or receive mails.

- Apache Tomcat server is a web server present in the form of standalone application.
- It is an open-source server which is available for all the users.



Write a program to

1) Create an interface as a connection interface
2) In above interface. Create three abstract methods Createstatement(), preparestatement(), preparecall().
3) Return type of CreateStatement() is statement interface, preparestatement(), is prepared statement, preparecall() is callable statement interface.
4) In statement interface, create two methods executeQuery() and executeUpdate().
5) Extend statement to preparedStatement to callable interface.
6) Create implementation classes for connection statement , prepared statement & callable statement interface & override executeUpdate() as well as execute query() method.
7) Create implementation class & override all the 3 methods.
8) In CreateStatement(), Create object of statement interface and return to respective method. Perform same task for prepare statement(), preparecall().
9) Create class DriverManager, which contains static method, call it as getConnection(). The return type of getConnection() is connection interface.
10) In getConnection(), create object of connection interface & return it to method.
11) Create a class application which consists of main method.
12) In main method, we get connection(), which will return connection object.

13) With the help of connection object, call createStatement(), which will return statement object
14) With the help of statement object, we call executeUpadate() & executeQuery() method.

Code:

Types of components

Every web supplication has to types of components

- Front end
- Backend

Front-end

- The components which are visible to the user, are referred as frontend components. It is also known as user interface.
- Programmers are using web technologies like HTML, CSS, JavaScript to build a front-end component.
- E.g.: Navigation bars, forms, Videos, etc.

Back-End

- The components which are hidden from the user, are referred as backend components.
- Programmers are using J2EE to build back-End components.
- E.g.: Database, Programming languages.

Note: Development of front-end as well as back-end component referred as full stack application development

Types of web Applications:

There are two types of web applications

- Static web application
- Dynamic web application

Static web application

- The application which consists fixed contend is referred as static web application.
- This type of applications contains only front-end components.
- E.g.: blogs.

Dynamic web application.

- The application which consists contend, gets changed as per user is referred as dynamic web application.
- This type of application contains front-end as well as back-end components.
- E.g.: Amazon, Flipkart.

Mobile Applications

- Mobile applications are those applications which can be downloaded and installed on the mobile.
- Web applications are those applications which can be used directly without downloading and installing it. We can use web browsers for using web application.
- Desktop application are those applications which are to be downloaded and installed on to the computer for using it.

Distributed application:

- The applications which are divided into several parts but they are connected with each other are referred as distributed application
- To develop these types of applications programmers are using cloud technology.

```
package programs;

import java.util.Random;

public class Sample {
        public static void main(String[] args) {
            Random random = new Random();
            int number = random.nextInt(1000);
            System.out.println("Before "+ number);

            if (number <1000) {
                number += 1000;
            }
            System.out.println(number);
        }
}
```

JAR File:

- **JAR** stands for java archive.
- It is a compressed version of java programs **.**
- Each JAR file consists .java file to read a program whereas .class file to execute a program.
- The main purpose of JAR file is sharing programs from one developer to other.
  E.g.: All the inbuild classes and interfaces like String class, Object class, Collection Framework interfaces are shared from the java developer to programmer, with the help of JAR file present in JRE system library.
- These JAR files are the collection of packages, which are also referred as external libraries.
- To perform J2EE programs, programmer require external libraries.

Maven Project:

- Maven is one of the project management tools, which can perform following tasks:
  - Generation of project report
  - Management of Dependencies
  - Management of plugins
- By default, in maven project programmers get **pom.xml** which represents project object module.
- This file is used to specify dependencies.
- Dependencies are used to represent collection of jar files.
- Programmers can find the dependencies on **mvnrepository.com** website
- To select the dependency, programmer have to check Vulnerability and usage provided on website
- To specify dependencies, make use of <dependencies> tag in **pom.xml** file

What is XML file ?

- XML stands for extensible markup language
- It is represented by tags.
- It is used to perform java configuration.
- Note: HTML is used to develop web pages.

Steps to update maven project:

- Right click on project → select maven option → maven → select update project.
- Select force update of snapshot and click on OK.

Steps to create maven project

- Search for maven project in IDE platform.
- Select create a simple project → click on next button.
- Specify group id and artifact if & then click on finish

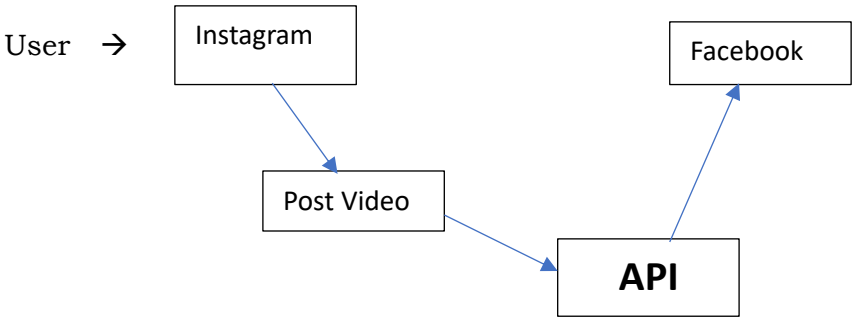
Internal working of maven project for dependencies:

- The dependencies which are specified in **pom.xml** file, will be accessible for maven project.
- To get jar files of respective dependency, maven will check local storage.
- If jar files are present in local storage, then maven will access them form its local storage.
- If jar files are not present in local storage then maven will download then from maven server & store them in a local storage

API

- API stands for Application programming interface
- It is the collection of classes and interfaces
- It is used for inter-Application communication which means sharing users' data from one application to other application.
- APIT is a mediator between applications.
- Each and every API is similar to packages.

E.g.: Java API ~ java.lang package
- Utility API ~ java.util package
- JDBC API ~ java.sql package
- Servlet API ~ javax.servlet package



Driver

Drivers are the translators, which will convert one language to another language.

E.g.:

Compiler : Compiler is a driver, which converts high level language to numerical language (Source code to byte code)

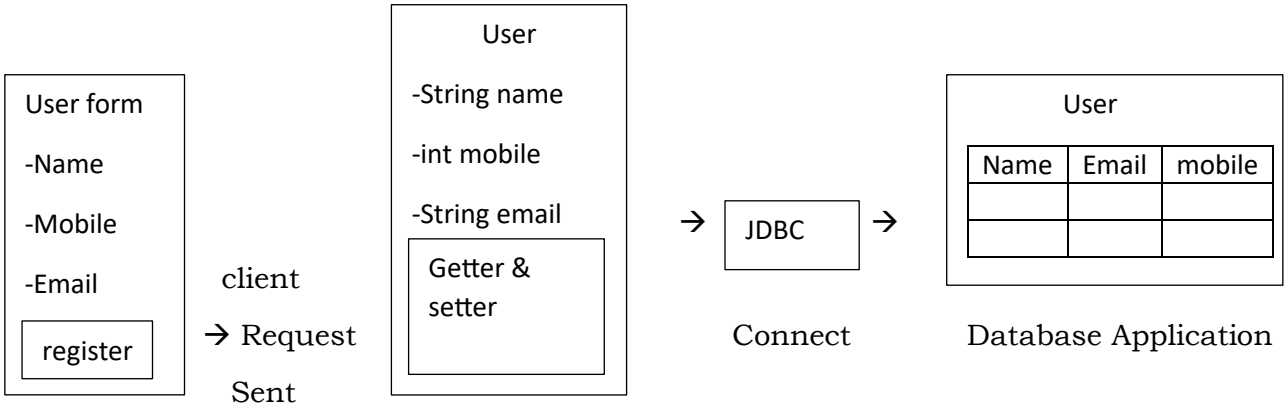
Interpreter: Interpreter is a driver, which converts numerical language to Binary level language (byte code to machine code)

Decompiler: Decompiler is a driver, which converts numerical language to source code (Byte code to Source code)

JDBC: Java Database Connectivity

As a programmer, we can't store users' data permanently in a java application.

Hence, store users' data from java application to database application by using java Database connectivity



JDBC Architecture:

There are 4 components in JDBC architecture

1. Java Application
2. JDBC API
3. JDBC Driver
4. Database Application

Java Application:

- It is used to write source code, to take input from user.
- Programmers can take input from 2 ways :
    - Scanner class
    - HTML file

JDBC API:

- It is used to share data from java application to database application.
- It is similar to java.sql package.
- It consists following classes and interfaces.

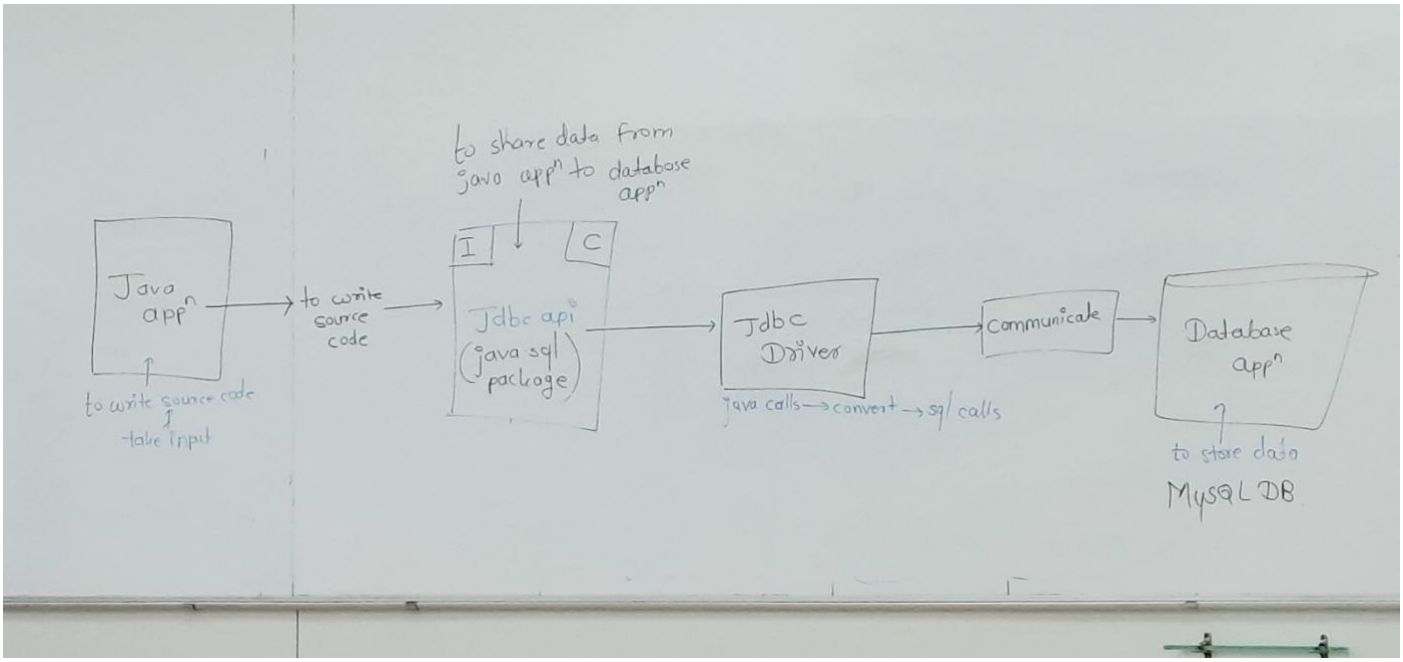
    i. DriverManager class
    ii. Connection interface.
    iii. Statement interface
    iv. PreparedStatement interface
    v. ResultSet interface
    vi. CallableStatement interface

JDBC Driver:

- It is used to convert java calls to SQL calls.
- JDBC API needs JDBC driver to communicate with database application.
- JDBC driver represented by using MySqlConnector Jar file.

Database Application:

It is used to store data permanently in the form of rows and columns.



URL:

URL stands for Uniform Resource Locator.

It is used to access database application in to java Application.

It contains database information like database name, protocol, host information, user information and port number.

Syntax for database URL:

protocol//hostinfo:portnumber/dbname?userinfo

E.g.: For MySQL database protocol is jdbc:mysql, post information is localhost, post number is 3307 and user information is username=root & password=Saurabh@9525

jdbc:protocol//localhost:3306//J2EEa3?user=root & password=Saurabh@9525

Steps of JDBC:

There are 6 steps of JDBC:

1. Load JDBC driver.
2. Established connection from java application to database application.
3. Create a platform
4. Execution of SQL Query.
5. Process of result set
6. Close the connection.

1. How to load JDBC Driver?

- Programmers are using forName method to load driver.
- ForName is a static method present in **Class** class.
- It takes an argument as fully qualified class name of Driver class i.e. com.mysql.cj.jdbc.Driver
- It throws an exception "ClassNotFound" Exception.
- e.g.:

```
package com.jdbc;

public class Step1 {
	public static void main(String[] args) {
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			System.out.println("done");
		} catch (ClassNotFoundException e) {

			e.printStackTrace();
		}
	}
}
```

2. How to establish connection between Java Application and database application?

- To establish the connection between java application and database, we have to use URL of respective database
- Programmers are using getConnection() static method present in DriverManager class.
- For this method, programmers provide argument as an URL of database application.
- It returns Connection interface type object.
- It throws an exception i.e. SQLException. It is a checked exception present in java.sql package
- It is a static overloaded method.

Example:

```
package com.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Step1 {
	public static void main(String[] args) {
```

```
            try {
                  // step-1
                  Class.forName("com.mysql.cj.jdbc.Driver");

                  // step-2
                  String  url  =  "jdbc:mysql://localhost:3306/j2ee?user=root  &
password=Saurabh@9525";
                  Connection connection = DriverManager.getConnection(url);
                  System.out.println("Connection Established...");
            } catch (ClassNotFoundException e) {
                  e.printStackTrace();
            } catch (SQLException e) {
                  e.printStackTrace();
            }
      }
}
```

```
package com.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;

public class Step1 {
      public static void main(String[] args) {
            try {
                  // step-1
                  Class.forName("com.mysql.cj.jdbc.Driver");

                  // step-2
                  String url = "jdbc:mysql://localhost:3306/j2ee";
                  String user = "root";
                  String password = "Saurabh@9525";
                  Connection connection = DriverManager.getConnection(url, user,
password);
                  System.out.println("Connection Established...");
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
}
```

3. What is platform and explain its type?

- “JDBC code contains java code as well as SQL queries, Java application can’t understand query language.
- Hence, programmers are writing queries in the form of string, to convert this string object into proper SQL queries, programmers are using JDBC Drivers.
- After conversion, to carry query from java application to database application, programmers are using platform.”

We have three types of platforms:

- Statement
- preparedStatement
- CallableStatement

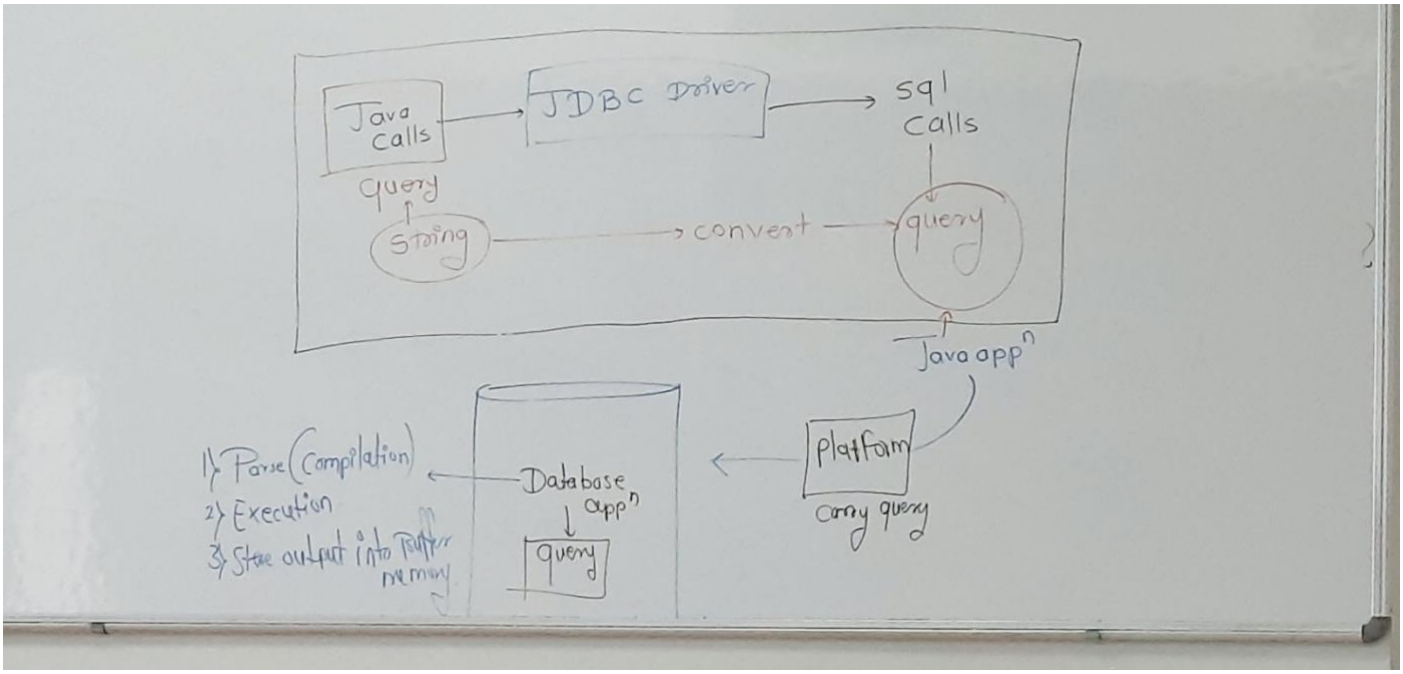
Statement platform is used to carry the queries in which programmer provided the data.

E.g.: insert into emp values(1, 'xyz', 1000);

The query which contains runtime values, to carry these queries programmers are using preparedStatement platform

E.g.: insert into emp values(?, ?, ?);

Programmers are using callableStatement platform to work with stored procedure



How to create statement platform?

- To create statement platform, programmer have to call **createStatement()** method present in **Connection** Interface
- Return type of **createStatement() is Statement** interface
- **E.g.:**

```
Statement statement = connection.createStatement();
```

```
package com.establish_connection;


import java.sql.Connection;
import java.sql.DriverManager;
```

```java
import java.sql.Statement;

public class EstablishConnection {

    public static void main(String[] args) {
        try {
            // step-1 : load JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // step-2 : Establish connection between java apk to database apk

            String url = "jdbc:mysql://localhost:3306/j2ee";
            String user = "root";
            String password = "Saurabh@9525";
            Connection connection = DriverManager.getConnection(url, user, password);
            System.out.println("Connection Established...");

            // step-3 : create  a platform
            Statement statement = connection.createStatement();
            System.out.println("Platform created...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. **How to execute SQL queries by using JDBC code?**

- Programmers can perform two operations on database application are as follows:
    - Write operation
    - Read operation
- To perform modification in database application programmers are using insert, update and delete queries (**DML**) is referred as **write operation.**
- To fetch data from database programmers are using select query (**DQL**) is referred as **Read operation.**
- To execute using insert, update and delete query, programmers prefer **"executeUpdate()"** method whereas to execute select query **executeQuery()** method.
- Above both the methods are present in statement interface.

Update the email :

```java
package com.establish_connection;
```

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class EstablishConnection {

    public static void main(String[] args) {
        try {
            // step-1 : load JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // step-2 : Establish connection between java apk to database apk

            String url = "jdbc:mysql://localhost:3306/j2ee";
            String user = "root";
            String password = "Saurabh@9525";
            Connection connection = DriverManager.getConnection(url, user, password);
            System.out.println("Connection Established...");

            // step-3 : create  a platform
            Statement statement = connection.createStatement();
            System.out.println("Platform created...");

            // step-4 : Execute the queries

           String query = "update employee set empemail='abc@email.com' where emoid=1";

            // return no. of records inserted/ deleted / updated
            int status = statement.executeUpdate(query);
            if(status !=0) {
                System.out.println("Email Updated...");
            }else {
                System.err.println("No data found...");
            }

                    } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```java
package com.establish_connection;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class EstablishConnection {

    public static void main(String[] args) {
        try {
            // step-1 : load JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // step-2 : Establish connection between java apk to database apk

            String url = "jdbc:mysql://localhost:3306/j2ee";
            String user = "root";
            String password = "Saurabh@9525";
            Connection connection = DriverManager.getConnection(url, user, password);
            System.out.println("Connection Established...");

            // step-3 : create  a platform
            Statement statement = connection.createStatement();
            System.out.println("Platform created...");

            // step-4 : Execute the queries

                                    String    query    =    "insert    into    employee
values(01,'Saurabh','test@gmail.com',70000)";
            statement.executeUpdate(query);
            System.out.println("Record Inserted...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```java
package com.establish_connection;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class EstablishConnection {

    public static void main(String[] args) {
        try {
            // step-1 : load JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // step-2 : Establish connection between java apk to database apk

            String url = "jdbc:mysql://localhost:3306/j2ee";
            String user = "root";
            String password = "Saurabh@9525";
            Connection connection = DriverManager.getConnection(url, user, password);
            System.out.println("Connection Established...");

            // step-3 : create  a platform
            Statement statement = connection.createStatement();
            System.out.println("Platform created...");

            // step-4 : Execute the queries

                                    String query = "insert into employee values(02,'Saurabh','test1@gmail.com',70000)";
                                   String query2 = "insert into employee values(03,'Mayur','mayur@gmail,com',74000)";
            statement.executeUpdate(query);
            statement.executeUpdate(query2);
            System.out.println("Record Inserted...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Question:

- Enter 1 to add employee.
- Enter 2 to remove employee if id=1
- Enter 3 to update employee name by its email.
- Enter 4 to remove employee if email is 'xyz@email.com'
- Enter 5 to update salary and name by its id.

```
package jdbc_application;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

public class Application {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            String url = "jdbc:mysql://localhost:3306/j2ee";
            String user = "root";
            String password = "Saurabh@9525";

            Connection con = DriverManager.getConnection(url, user, password);

            Statement stmt = con.createStatement();
            boolean t = true;
            while (t) {
                System.out.println("Enter Choice to perform the following tasks:");
                System.out.println("1.  Add Employee ");
                System.out.println("2.  Remove Employee with id 1");
                System.out.println("3.  Update Employee name by its email");
                System.out.println("4.  Remove Employee if email is : xyz@email.com");
                System.out.println("5.  Update salary and name of Employee by its id");

                int choice = sc.nextInt();
                int status = 0;

                switch (choice) {
                case 1:
                                                String q1 = "insert into employee
values(06,'Vikaass','Vikaaas@gmail.com',71000)";
                    stmt.executeUpdate(q1);
                    System.out.println("Employee added...");
                    break;



                case 2:
                    String q2 = "delete from employee where emoid=1";
```

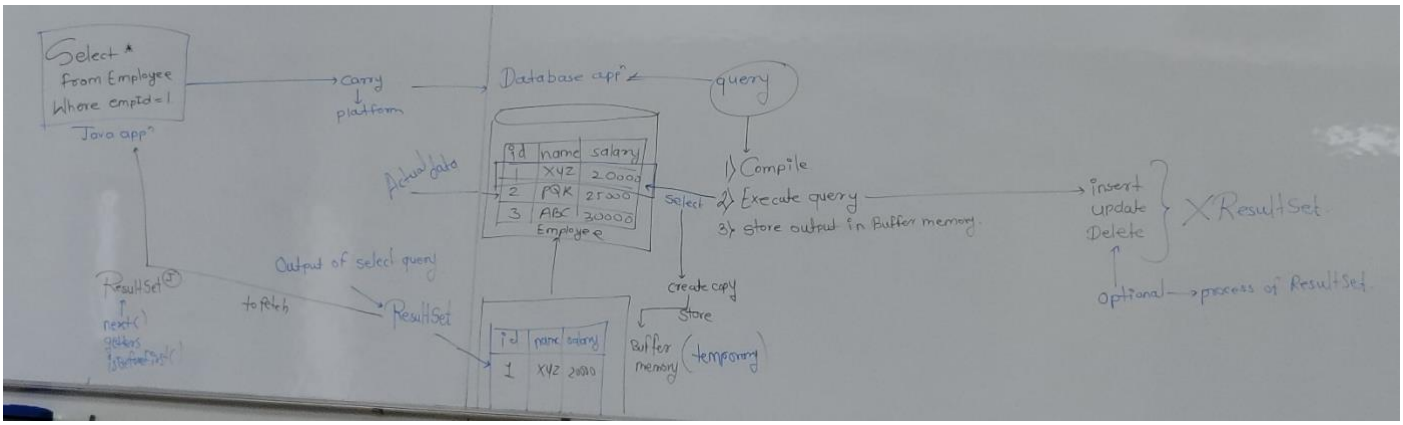
```
                    status = stmt.executeUpdate(q2);
                    if (status != 0) {
                        System.out.println("Employee deleted...");
                    } else {
                        System.out.println("Error");
                    }
                    break;
                case 3:
                    String q3 = "update employee set empname = empemail where emoid=2";
                    stmt.executeUpdate(q3);
                    System.out.println("Employee name updated...");
                    break;
                case 4:
                    String q4 = "delete from employee where empemail='mayur@gmail.com'";
                    status = stmt.executeUpdate(q4);
                    if (status != 0) {
                        System.out.println("Employee deleted...");
                    } else {
                        System.err.println("Error...");
                    }

                    break;
                case 5:
                    System.out.println("enter Id to update employee : ");
                    int id = sc.nextInt();
                     String q5 = "update employee set salary = 70000  , empname = 'Sajjad'
where emoid=" + id + "";
                    status = stmt.executeUpdate(q5);
                    if (status != 0) {
                        System.out.println("Employee updated...");
                    } else {
                        System.err.println("Error...");
                    }

                    break;
                case 6:
                    System.out.println("enter  6 for exit ");
                    t = false;
                    break;
                default:
                    System.err.println("Something went wrong...");
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

5. Process of result set:



Verify email:

```
package com.jsp.verify;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class VerifyEmail {
	public static void main(String[] args) {
		try {
			// load driver
			Class.forName("com.mysql.cj.jdbc.Driver");
			String url = "jdbc:mysql://localhost:3306/j2ee";
			String user = "root";
			String password = "Saurabh@9525";

			// create connection
			Connection con = DriverManager.getConnection(url, user,
			 password);

			// create platform
			Statement stmt = con.createStatement();

			// execute queries
			String query = "select * from employee where
empemail='test1@gmail.com'";

			ResultSet rs = stmt.executeQuery(query);
			// process result set
			boolean status = rs.next();
			if (status) {
				System.out.println("Email verified...");
			} else {
				System.err.println("Error...");
			}

			// close the connection
			con.close();

		} catch (Exception e) {

			e.printStackTrace();
		}
```

```
	}
}
```

```
Email verified...
```

Check whether Given name and id is present in database or not

```
package com.jsp.verify;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Scanner;

public class Check {
	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);

	try {
		// load driver
		Class.forName("com.mysql.cj.jdbc.Driver");
		String url = "jdbc:mysql://localhost:3306/j2ee";
		String user = "root";
		String password = "Saurabh@9525";

		// create connection
		Connection con = DriverManager.getConnection(url, user, password);

		// create platform
		Statement stmt = con.createStatement();

		// execute queries
		System.out.println("Enter name and id for searching... ");
		String name = sc.next();
		int id = sc.nextInt();

		String query = "select * from employee where empname='"+ name + "'
and emoid ="+ id + "";

		ResultSet rs = stmt.executeQuery(query);
		// process result set
		boolean status = rs.next();
		if (status) {
			System.out.println("User Present in database...");
		} else {
			System.err.println("Error...");
		}

		// close the connection
		con.close();

	} catch (Exception e) {

		e.printStackTrace();
	}
```

```
}
}
```

```
Enter name and id for searching...
Sajjad
6
User Present in database...
```

```
package com.jsp.verify;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Scanner;

public class Check {
	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);

	try {
		// load driver
		Class.forName("com.mysql.cj.jdbc.Driver");
		String url = "jdbc:mysql://localhost:3306/j2ee";
		String user = "root";
		String password = "Saurabh@9525";

		// create connection
		Connection con = DriverManager.getConnection(url, user, password);

		// create platform
		Statement stmt = con.createStatement();

		// execute queries
		System.out.println("Enter name and id for searching... ");
		String name = sc.next();
		int id = sc.nextInt();

		String query = "select * from employee where empname='"+ name + "'
and emoid ="+ id + "";

		ResultSet rs = stmt.executeQuery(query);
		// process result set

		if (rs.next()) {
			System.out.println("User Present in database...");
			String ename = rs.getString("empname");
			System.out.println("Name = "+ ename);
			int eid = rs.getInt("emoid");
			System.out.println("ID = "+ eid);
			double sal = rs.getDouble("salary");
			System.out.println("Salary = "+ sal);
		} else {
			System.err.println("No data found...");
		}
```

```
            // close the connection
            con.close();
            sc.close();

      } catch (Exception e) {

            e.printStackTrace();
      }

}
}
```

```
Enter name and id for searching...
Sajjad
6
User Present in database...
Name = Sajjad
ID = 6
Salary = 70000.0
```

What is result Set?

- It is an output of select query
- When database executes select query it will store copy of respective record in a buffer memory, is referred as ResultSet.
- For insert, update and delete query, database does not generate any result set.
- To get the result set from buffer memory, in java application programmer prefers, **ResultSet** interface present in **java.sql** package.
- In contains three methods:
    - **next()**
    - **getters()**
    - **isBerforeFisrt()**
- The **next()** method is used to shift the cursor from one position to another as well as it will check whether at current position record is present or not.
- Return type of this method is Boolean.
- To fetch the data from ResultSet programmers are using **getters()** method.
- To check whether first record present in ResultSet or not, programmers are using **isBeforeFirst()** method.
- ResultSet interface is a return type of **executeQuery()** method.

What is prepared statement

- It is a platform which is used by programmer to work with runtime values.
- To specify runtime values in query, programmers are using placeholder which is denoted with question mark.
- Prepared statement compiles query only once whereas executes multiple times.
- To create prepared statement platform programmers are using **preparedStatement()** method, present in **Connection** interface.
- The return type of this method is **preparedStatement** Interface present in **java.sql** package
- e.g.:

```
String query = "insert into book(name, price, pages) values (?,?,?)";
PreparedStatement pstmt = con.prepareStatement(query);
```

- To assign the values to placeholder programmers are using setter method respective with datatype of value.
- e.g.:

```
String name = sc.next();
pstmt.setString(1, name);
```

What is difference between statement and prepared statement

Sr.No.	statement	prepared statement
1	It is used to execute the queries which consists values, given by programmer	It is used to execute query which consists values given by user
2	It compiles and execute query multiple times	It compiles query only one time but execute multiple times
3	Programmer can't pass placeholders to work with runtime values	Programmers are using placeholders to take runtime values
4	It is a base interface	It extends statement interface
5	It executes normal SQL query	It executes dynamic SQL queries
6	It is used to execute DDL statements	It is used to execute DML and DQL statements

What is callable statement?

- it is used to work with stored procedure.
- Stored procedures are the methods which will save SQL queries permanently in a database application.
- It is set of multiple sql statement.
- We can create stored procedure with as well as without parameter.

E.g.:

1) **Stored procedure without parameter**

   CREATE PROCEDURE `new_procedure` ()

   BEGIN

   INSERT INTO book(bookName, Price, Pages) values ('Java', 240,758);

   END

2) **Stored procedure with parameter**

   CREATE PROCEDURE `new_procedure` (in bprice double)

   BEGIN

   SELECT * from book where price > bprice;

   END

To call stored procedure from java application, use below syntax:

Call StoredProcedureName

E.g.: To call stored procedure without parameter:

String callSP = “call insertData()”;

To call stored procedure with parameter

String callSp = “call FindBookGreaterThanGivenPrice(?)”;

To create callable statement, programmers are using “PrepareCall()” method present in Connection interface

The return type of given method Is callable statement interface.

E.g.: CallableStatement cstmt = con.prepareCall(callSP);

Servlet:

Servlet Interface – javax.Servlet

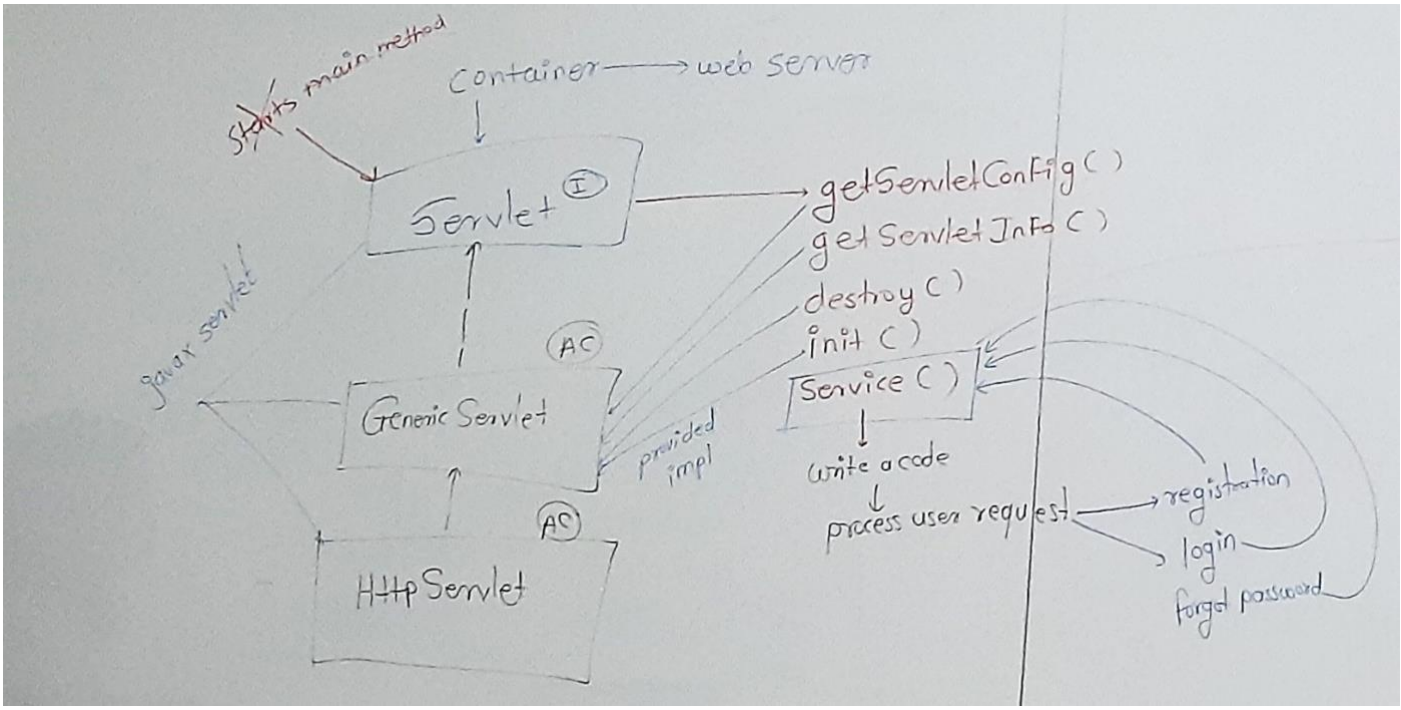
- To develop web application
- To develop dynamic web pages
- Server side programming language
- To connect front end to the backend
- Two features
    - Multi-threaded
    - Session

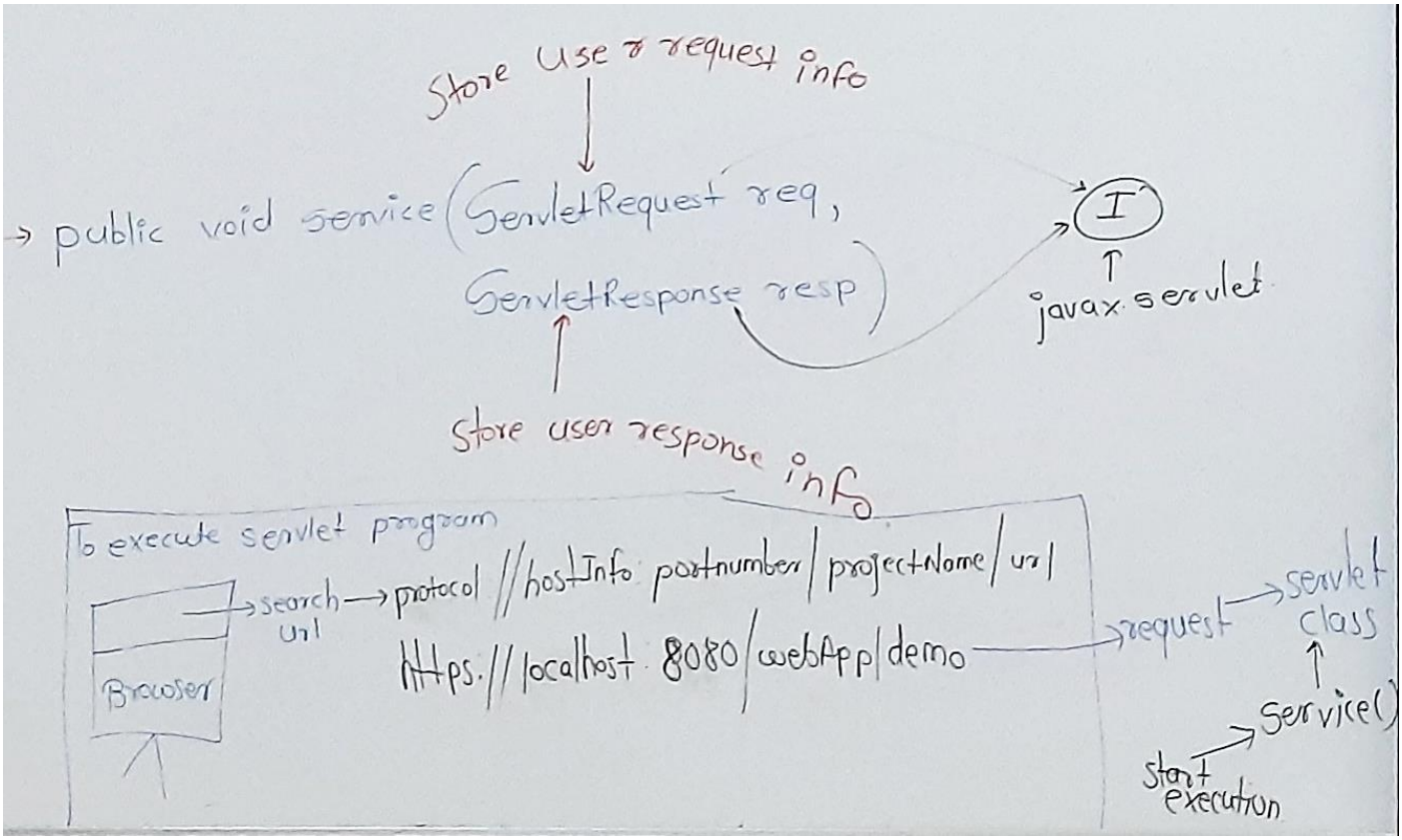
Multi-threaded:

Multiple users can access at the same time.

Session:

It is a particular time of interval given by programmer to the user to perform specific task





Servlet chaining

↓

Request Dispatcher

↓

To transfer data → Servlet Class

Servlet JSP

ServletRequest (I)

↓

setAttributes(indentifier, data)

req.setAttributes("number", 10)

↳ upcasted to **Object Class**

int – Integer

double – Double

to access data in servlet class / jsp file

ServletRequest (I) → getAttributes("identifier")

↳ Object Class

What is servlet ?

- Servlet is an interface present in javax.servlet package
- It is a container present in server.
- It is used to develop dynamic web pages as well as web applications.
- It is used to connect front end of an application with backend of an application.
- We have two types of servlet classes
    - Generic servlet
    - Http Servlet
- Servlet interface consists following methods.
    - Init()
    - Destroy()
    - getServletInfo()
    - getServletConfig()
    - service()
- Generic servlet implements - Init() , destroy(), getServletConfig(), getServletInfo() methods from servlet interface
- Service() method is not implemented because it is used to write code to handle user request
- HttpServlet class extends generic servlet
- Note: javax.Servlet package is not present in JRE system library hence programmer have to provide dependency of servlet API.
- By using servlets programmers can provide two features for an application
    - Multi-threaded
    - Session

What is generic Servlet ?

- It is an abstract class present in javax.servlet package
- It contains an abstract method service()
- For service method there are two arguments –
    - ServletRequest res
    - ServletResponse req
- The main purpose of Generic servlet is to provide multithreaded feature for an application
- Generic servlet is protocol independent which means by using this servlet programmers can process FTP(File Transfer Protocol) , TCP (Transmission Control Protocol) request.

How to create Generic Servlet class ?

- Create a normal class in a java package
- Extends generic servlet class in a current class from javax.servlet package
- Override service() method from generic servlet class and write a business logic of an application.

How to start the execution of servlet classes ?

- Programmers can't execute servlet class by using main() method.
- All the servlet classes are present in a server.
- To send the request to the server programmers are using browsers.
- To start execution of servlet class programmers can send request, by using HTML form or by using Hyperlink or programmer can directly search URL of Servlet in URL bar.
- Syntax for servlet URL : http://localhost:8080/project_name/url_of_servlet

Explain @WebServlet Annotation

- It is used to map URL with respective servlet class
- It is specified above Servlet class name.
- Ex. @WebServlet("/Map_Name")

Explain PrintWriter class.

- It is used to print java code output on the browser
- It is present in java.io package
- To get the Object of PrintWriter class, make use- of getWriter() method present in ServletResponse Interface.
- Then call Println() method, to print an output.
- To enable HTML tags, make use of setContentType() method present in ServletResponse Interface and assign an argument as **"TEXT/HTML".**

How to fetch data from URL or HTML form in Servlet class ?

- To fetch the data from URL or HTML form, programmers are using getParameter() method present in ServletRequest Interface.
- For above method provide argument as key of URL parameter or name of input box.
- Return type of above method is string.

What is URL parameter ?

- It is used to take input from an user while sending get request, it is present in the form of key-value pair.
- Programmer can provide this URL parameters by using HTML form with input box.
- **Syntax:** http://localhost:8080/demo?key1=value1&key2=value2
- While working with HTML form name of input box represented as a key and data entered in input box represented as value.

How to send request for servlet class by using HTML form ?

- To send the request by using HTML form, programmers are using action attribute with form tag and provide value as Servlet class map URL.
- Then create submit button in a HTML form.

What is HttpServlet ?

- It is an abstract class present in javax.servlet.http package.
- It does not contains any abstract methods.
- In HttpServlet class, programmers implemented service() method and introduce new methods like doGet(), doPost(), doPut(), doOption(), doTrace().
- Above all the methods are protected and have two arguments –
    - HttpServlet req
    - HttpServlet resp
- HttpServlet supports multi-threaded as well as session feature of an application.
- It is protocol dependent which means it will process only HTTP request(Hypertext Transfer Protocol).

Explain difference between get and post request?

Get	Post
1. By default, user sends post request to servlet, when user click on submit button or user search the URL of servlet or user click on the hyperlink.	1. If programmer wants to send post request, then programmer prefers html form, which means programmer will use method attribute with form tag.
2. Get request is not a secured request	2. Post is a secured request
3. Get request displays user's data in URL	3. Post request Hides user's data in URL
4. We can send less information using get request because, data is transferred in URL	4. By using post request we can send large amount of information because data is not sent by using URL

What is the difference betn

GenericServlet	HttpServlet
1. GenericServlet implements servlet interface	HttpServlet extends Generic servlet
2. GenericServlet is protocol independant	HttpServlet is Protocol dependant
3. Generic servlet supports only multithreaded feature	HttpServlet supports multithreaded as well as session feature
4. Generic servlet does not support sendRedirect method	HttpServlet supports sendRedirect method
5. In generic servlet service method is an abstract method	In HttpServlet service method is a complete method.
6. GenericServlet Does not support Sessions	HttpServlet supports Sessions

What is servlet chaining?

- When one servlet class will communicate with other resources like html file
  - ,jsp file, other servlet classes is referred as servlet chaining.
- The main purpose of servlet chaining is linking one functionality of an application to other functionality for sequential flow of execution.
- Programmer have two approaches to perform servlet chaining.
  - By using RequestDispatcher interface
  - By using sendRedirect method

Explain RequestDispatcher interface

Ans:

- It is used to perform servlet chaining.
- It is present in javax.servlet package.
- It can communicate with the resources present in a project.
- The helper method of RequestDispatcher is “getRequestDispatcher()”.
- Above method is present in ServletRequest interface.
- To transfer request and response object, it contains two methods.
    - Include()
    - Forward()
- While performing servlet chaining programmer can transfer the data from one servlet class to JSP file or another servlet class.
- To perform above operation, programmers are using “setAttribute” method present in ServletRequest interface.
- For above method programmer have to provide two arguments
    - Identify
    - Data
- To access above information into another servlet class programmer prefers getAttribute method present in ServletRequest interface.
- For above method provide argument as identify. The return type of this method is object class hence programmer have to perform down casting.

What is sendRedirect() method ?

- It is used to perform servlet chaining with the resiources resent in the project or ottside the project.
- Programmers can use this sendRedirect() method only with HttpServlet class
- This method is present in HttpServlet Interface
- It provides final response to the user.
- It will change the url for redirected request.
- Programmers can provide arguments as Html file name, JSP file name and Servlet class URL as well as URL of external resources.

Write a difference between include and forward method.

Include	Forward
Include method combines multiple resources output	Forward will transfer request from servlet class to other resources without including a response.
If programmers wants response from multiple servlet classes then programmers prefers include method	If programmer wants to give final response to the user then programmer prefer forward method.
Programmer can use multiple include method in a servlet	Programmer can use forward only once.

How to transfer data from one servlet to another servlet or JSP file at the time of servlet chaining ?

Ans:

- To transfer data from servlet to another servlet or JSP file, programmers prefer "SetAttribute()" method.
- For above method programmer have to pass 2 arguments,
    - Identifier
    - Data
- Set attribute method supports the servlet chaining performed by using RequestDispatcher interface.
- It will upcast data to object class.
- To access transferred data in requested resources programmers are using "getAttribute()" method.
- setAttribute() and getAttribute(), both the methods are present in ServletRequest interface.

What is session?

Ans:

- Session is a time interval provided by programmer to the user for performing a respective task.
- It is also a time interval between login and logout.
- It is an object, created by web container and present in server.
- When user logged in an application, then programmer have to collect user data from database and store it in a session object.
- If in further process, programmer requires user data then he will access it form session instead of database .
- To access session object, call getSession() method from HttpServlet Request Interface, it returns HttpSession type object.
- HttpSession is an interface.
- To store the data in a session object, make use of setAttribute() method present in HttpSession Interface.
- To access this data in a servlet class, make use of getAttribute() method present in HttpSession Interface.
- To provide the time interval, make use of setMaxInactiveInterval() method.
- To close the session make use invalidate() method.
- Programmers can provide session feature for an application by using HttpServlet class.
- After completion of time interval, data will get removed from session object.

Explain JSP.

Ans:

- It stands for Java Server Pages.
- It is a extension of Servlet.
- It is used to develop dynamic web pages.
- By default JSP file consists java code.
- We can write the java code in JSP file by using Scriplets (<% … %>)
- We can print output of the java code on the browser by using expression (<%= %>)

```
<%
    List<Employee> emp = (List)
request.getAttribute("EmpData");
    if (emp != null) {
    %>
    }
    <form action="">

        <%
            For(Employee e1 : emp){
        %>
        <label>Employee Id :</label>
        <label>Employee Name :</label>
        <label>Employee Email :</label>
        <label>Employee Salary :</label>
        <label>Employee Mobile :</label>
        <%
        }
        %>

        </form>
```