**INDORE INSTITUTE OF SCIENCE & TECHNOLOGY, INDORE**

Department of Computer Science & Engineering



**LAB MANUAL**

**Subject: - Programming Practices (CS-406)**

**B.Tech: Second Year/ IV SEM**

**Name: Neelesh Singh**

**Enrollment no: 0818CS201108**

**Section: - CS2**

**Session 2021-22**

#### EXPERIMENT NO.1

**Aim / Title:** Write a program to implement ArrayList and Vector in Collection Framework.

**Problem Statement:** Write a program to implement List interface

**Objectives:** Understand and implement List interface of Collection

**Outcomes:** Student will be able to apply and develop applications on List interface

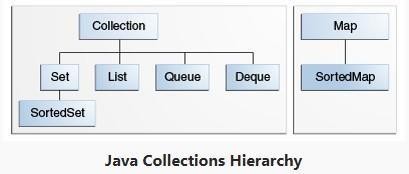
**Pre-requisite:** Basic Knowledge of Java

**Hardware requirements:** Processor: 1 (GHz) RAM: 2 gigabyte (GB) for Hard disk space: 16 GB .Graphics card: DirectX 9 or later with WDDM 1.0 driver. Display: 800x600.

#### Software requirements: JDK

**Theory**:

The Java Collections Framework is a collection of interfaces and classes which helps in storing and processing the data efficiently. This framework has several useful classes which have tons of useful functions which makes a programmer task super easy. I have written several tutorials on Collections in Java.



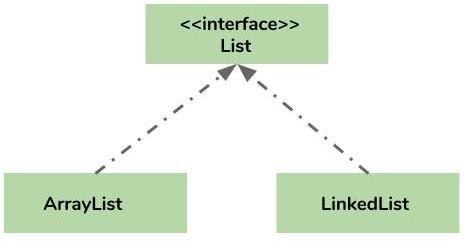
#### List

The Java.util.List is a child interface of [Collection.](https://www.geeksforgeeks.org/collections-in-java-2/) It is an ordered collection of objects in which duplicate values can be stored. Since List preserves the insertion order, it allows positional access and insertion of elements. List Interface is implemented by the classes of [ArrayList,](https://www.geeksforgeeks.org/arraylist-in-java/) [LinkedList,](https://www.geeksforgeeks.org/linked-list-in-java/) [Vector](https://www.geeksforgeeks.org/java-util-vector-class-java/) and [Stack.](https://www.geeksforgeeks.org/stack-class-in-java/)

##### public interface List<E> extends Collection<E>

**ArrayList** class implements List interface and it is based on an Array data structure. It is widely used because of the functionality and flexibility it offers. Most of the developers choose ArrayList over Array as it’s a very good alternative of traditional java arrays. ArrayList is a

resizable-array implementation of the List interface. It implements all optional list operations,



and

permits all elements, including null.

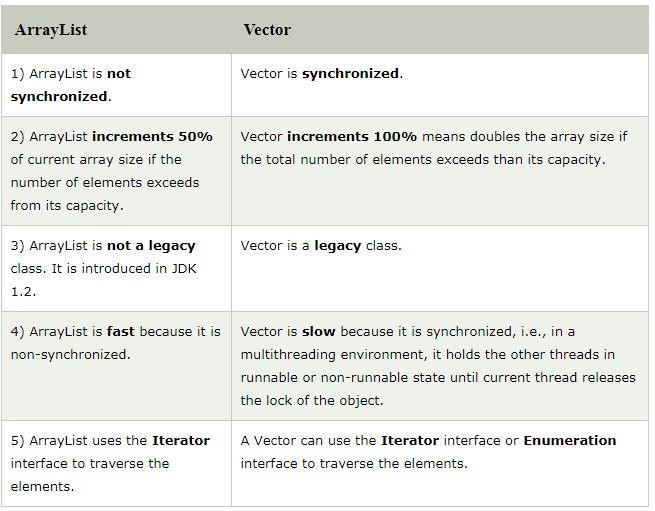
The limitation with array is that it has a fixed length so if it is full you cannot add any more elements to it, likewise if there are number of elements gets removed from it the memory consumption would be the same as it doesn’t shrink.

We can create an ArrayList by writing a simple statement like this:

*ArrayList<String> alist=new ArrayList<String>(); ArrayList<Integer> list=new ArrayList<Integer>();*

#### VECTOR

Vector implements List Interface. Like ArrayList it also maintains insertion order but it is rarely used in non- thread environment as it is synchronized and due to which it gives poor performance in searching, adding, delete and update of its elements.



Three ways to create vector class object:

1. Vector vec = new Vector(); 2. Vector vec = new Vector(3); 3. Vector vec= new Vector(4, 6)

#### Program:

**import java.util.\*;**

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

/\* Creating ArrayList of type "String" which means

\* we can only add "String" elements

\*/

ArrayList<String> obj = new ArrayList<String>();

/\*This is how we add elements to an ArrayList\*/ obj.add("Ajeet"); obj.add("Harry"); obj.add("Chaitanya");

obj.add("Steve"); obj.add("Anuj");

// Displaying elements

System.out.println("Original ArrayList:");

for(String str:obj)

System.out.println(str);

/\* Add element at the given index

obj.add(0, "Rahul") - Adding element "Rahul" at first position

obj.add(1, "Justin") - Adding element "Justin" at second position

\*/

obj.add(0, "Rahul");

obj.add(1, "Justin");

// Displaying elements

System.out.println("ArrayList for(String str:obj)

System.out.println(str);

after

add operation:");

//Remove elements from ArrayList like this obj.remove("Chaitanya"); //Removes "Chaitanya" from ArrayList

obj.remove("Harry"); //Removes "Harry" from ArrayList

// Displaying elements

System.out.println("ArrayList after remove operation:"); for(String str:obj)

System.out.println(str);

//Remove element from the specified index obj.remove(1); //Removes Second element from the List

// Displaying elements

System.out.println("Final ArrayList:"); for(String str:obj) System.out.println(str); } }

#### Vectors: import java.util.\*;

public class VectorExample {

public static void main(String args[]) {

/\* Vector of initial capacity(size) of 2 \*/ Vector<String> vec = new Vector<String>(2);

/\* Adding elements to a vector\*/ vec.addElement("Apple"); vec.addElement("Orange"); vec.addElement("Mango"); vec.addElement("Fig");

/\* check size and capacityIncrement\*/ System.out.println("Size is: "+vec.size());

System.out.println("Default capacity increment is: "+vec.capacity());

vec.addElement("fruit1"); vec.addElement("fruit2"); vec.addElement("fruit3");

vector.remove("Harry");

// Removing Matt vector.remove("Matt");

/\*size and capacityIncrement after two insertions\*/ System.out.println("Size after addition: "+vec.size()); System.out.println("Capacity after increment is: "+vec.capacity());

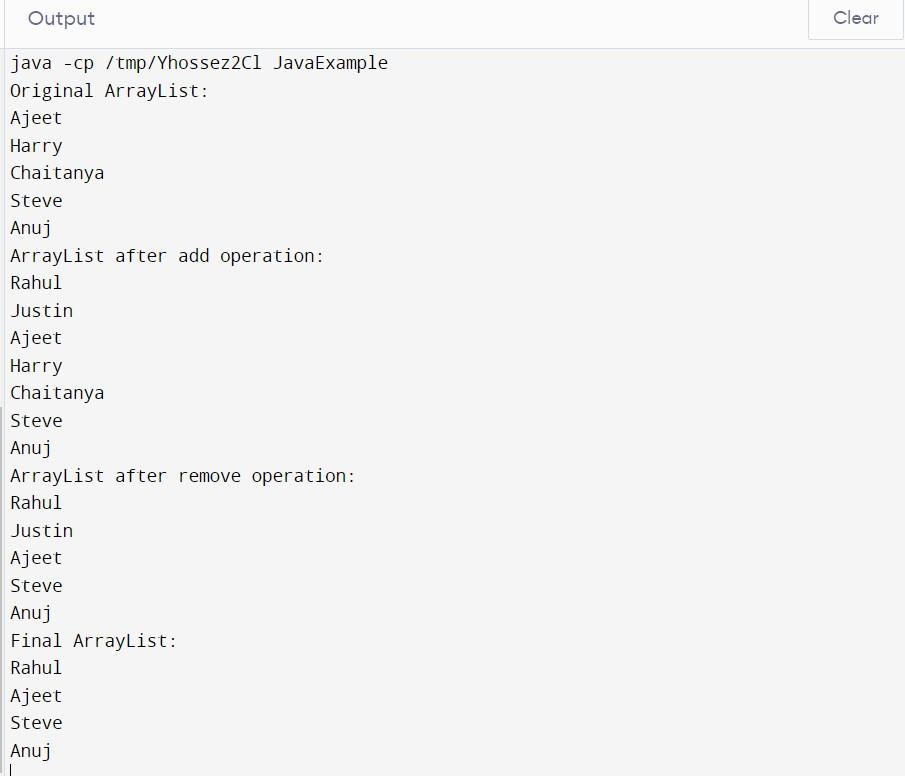
/\*Display Vector elements\*/ Enumeration en = vec.elements();

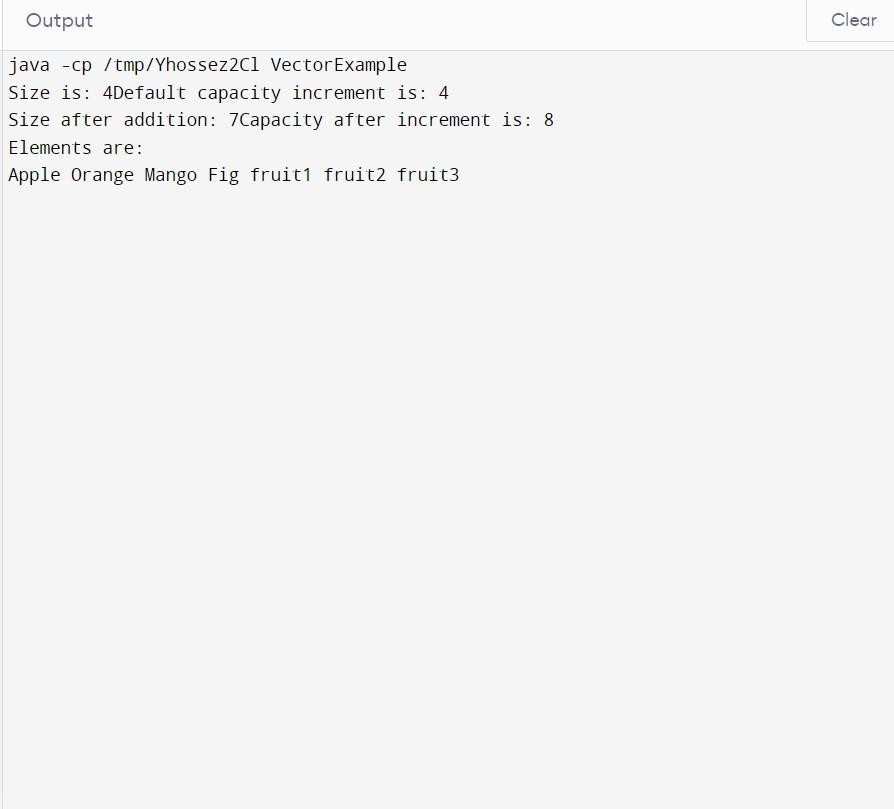
System.out.println("\nElements are:"); while(en.hasMoreElements())

System.out.print(en.nextElement() + " ");

int first\_index = vector.indexOf("Patt"); System.out.println("First Occurrence of Patt at index: "+first\_index);

} }





#### Conclusion:

The List interface in Java provides a way to store the ordered collection. It is a child interface of Collection. It is an ordered collection of objects in which duplicate values can be stored. Since List preserves the insertion order, it allows positional access and insertion of elements. The implementation classes of the List interface are ArrayList, LinkedList.

Sample Viva Questions and

#### Q1. What is Collection Framework?

Ans. The Java collections framework provides a set of interfaces and classes to implement various data structures and algorithms. For example, the LinkedList class of the collection’s framework provides the implementation of the doubly-linked list data structure.

#### Q2. What is the application of Collection Framework?

Ans. Java Collection Framework enables the user to perform various data manipulation operations like storing data, searching, sorting, insertion, deletion, and updating of data on the group of elements.

Iterator(): Return an iterator over collection Clear(): Removes all elements from the collection Add(): Add objects to collection

#### Q3. Difference between ArrayList and Vector?

Ans. ArrayList is non-synchronized. Vector is synchronized. ArrayList increments 50% of its current size if element added exceeds its capacity. Vector increments 100% of its current size if element added exceeds its capacity.

#### Q4. What is synchronous and asynchronous Lists?

Ans. Asynchronous functions are generally preferred over synchronous functions as they do not block the execution of the program whereas synchronous functions block the execution of the program until it has finished processing.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Roll No.** | **Name**  **Student** | **of** | **Date of**  **Performance** | **Date of**  **Evaluation** | **Grade** | **Sign**  **Student** | **of** | **Sign of**  **Faculty** |
| 0818CS  201108 | Neelesh Singh | |  |  |  |  | |  |

#### EXPERIMENT NO.2

**Aim / Title:** Write a program to implement HashSet and TreeSet in Collection Framework.

**Problem Statement:** Write a program to implement Set interface

**Objectives:** Understand and implement Set interface of Collection

**Outcomes:** Student will be able to apply and develop applications on Set interface

**Pre-requisite:** Basic Knowledge of Java

**Hardware requirements:** Processor: 1 (GHz) RAM: 2 gigabyte (GB) for Hard disk space: 16 GB .Graphics card: DirectX 9 or later with WDDM 1.0 driver. Display: 800x600.

#### Software requirements: JDK

**Theory**:

##### Set

Set is an interface which extends Collection. It is an unordered collection of objects in which duplicate values cannot be stored. Basically, Set is implemented by HashSet, LinkedHashSet or TreeSet (sorted representation). Set has various methods to add, remove clear, size, etc to enhance the usage of this interface. Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ.

#### HashSet Class

This class implements the Set interface, backed by a hash table (actually a HashMap instance). It makes no guarantees as to the iteration order of the set; in particular, it does not guarantee that the order will remain constant over time. This class permits the null element. This class is not synchronized.

* 1. HashSet doesn’t maintain any order, the elements would be returned in any random order.
  2. HashSet doesn’t allow duplicates. If you try to add a duplicate element in HashSet, the old value would be overwritten.
  3. HashSet allows null values however if you insert more than one nulls it would still return only one null value.
  4. HashSet is non-synchronized.
  5. The iterator returned by this class is fail-fast which means iterator would throw ConcurrentModificationException if HashSet has been modified after creation of iterator, by any means except iterator’s own remove method.

#### TreeMap Class

TreeSet is one of the most important implementations of the SortedSet interface in Java that uses a Tree for storage. The ordering of the elements is maintained by a set using their natural ordering whether or not an explicit comparator is provided. This must be consistent with equals if it is to correctly implement the Set interface. It can also be ordered by a Comparator provided at set creation time, depending on which constructor is used. The TreeSet implements a NavigableSet interface by inheriting AbstractSet class.

Few important features of TreeSet are as follows:

1. TreeSet implements the [SortedSet](https://www.geeksforgeeks.org/sortedset-java-examples/) interface so duplicate values are not allowed.
2. Objects in a TreeSet are stored in a sorted and ascending order.
3. TreeSet does not preserve the insertion order of elements but elements are sorted by keys.
4. TreeSet does not allow to insert Heterogeneous objects. It will throw classCastException at Runtime if trying to add hetrogeneous objects.
5. TreeSet serves as an excellent choice for storing large amounts of sorted information which are supposed to be accessed quickly because of its faster access and retrieval time.
6. TreeSet is basically implementation of a self-balancing binary search tree like [Red-Black](https://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/) [Tree.](https://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/) Therefore operations like add, remove and search take O(Log n) time. And operations like printing n elements in sorted order takes O(n) time.

#### Difference between HashSet and TreeSet

1. **Ordering :** HashSet stores the object in random order . There is no guarantee that the element we inserted first in the HashSet will be printed first in the output . Elements are sorted according to the natural ordering of its elements in TreeSet. If the objects cannot be sorted in natural order than use [compareTo() method to sort the elements](https://javahungry.blogspot.com/2013/08/difference-between-comparable-and.html) of TreeSet object.
2. **Null value :** HashSet can store null object while TreeSet does not allow null object. If one try to store null object in TreeSet object , it will throw Null Pointer Exception.
3. **Performance :** HashSet take constant time performance for the basic operations like add, remove contains and size.While TreeSet guarantees log(n) time cost for the basic operations (add,remove,contains).
4. **Speed :** HashSet is much faster than TreeSet,as performance time of HashSet is constant against the log time of TreeSet for most operations (add,remove ,contains and size) . Iteration performance of HashSet mainly depends on the load factor and initial capacity parameters.
5. **Internal implementation :** As we have already discussed [How hashset internally works in java t](https://javahungry.blogspot.com/2013/08/how-sets-are-implemented-internally-in.html)hus, in one line HashSet are internally backed by hashmap. While TreeSet is backed by a Navigable TreeMap.
6. **Functionality :** TreeSet is rich in functionality as compare to HashSet. Functions like pollFirst(),pollLast(),first(),last(),ceiling(),lower() etc. makes TreeSet easier to use than HashSet.
7. **Comparision :** HashSet uses equals() method for comparison in java while TreeSet uses compareTo() method for maintaining ordering .

#### Program: TreeSet import java.util.TreeSet;

|  |
| --- |
| public class TreeSetExample { public static void main(String args[]) {  // TreeSet of String Type  TreeSet<String> tset = new TreeSet<String>();  // Adding elements to TreeSet<String> tset.add("ABC"); tset.add("String"); tset.add("Test");  tset.add("Pen"); tset.add("Ink"); tset.add("Jack");  //Displaying TreeSet System.out.println(tset);  // TreeSet of Integer Type  TreeSet<Integer> tset2 = new TreeSet<Integer>();  // Adding elements to TreeSet<Integer> tset2.add(88); tset2.add(7); tset2.add(101); tset2.add(0); tset2.add(3); tset2.add(222);  System.out.println(tset2);  }  } |
| **HashSet** |

**import java.util.HashSet;**

public class HashSetExample {

public static void main(String args[]) {

// HashSet declaration

HashSet<String>

HashSet<String>();

hset

= new

// Adding elements to the HashSet

hset.add("Apple");

hset.add("Mango");

hset.add("Grapes");

hset.add("Orange");

hset.add("Fig");

//Addition of duplicate elements hset.add("Apple");

hset.add("Mango");

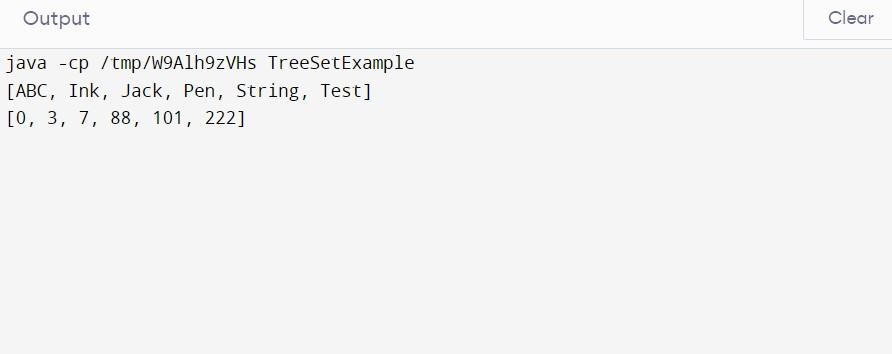
//Addition of null values hset.add(null); hset.add(null);

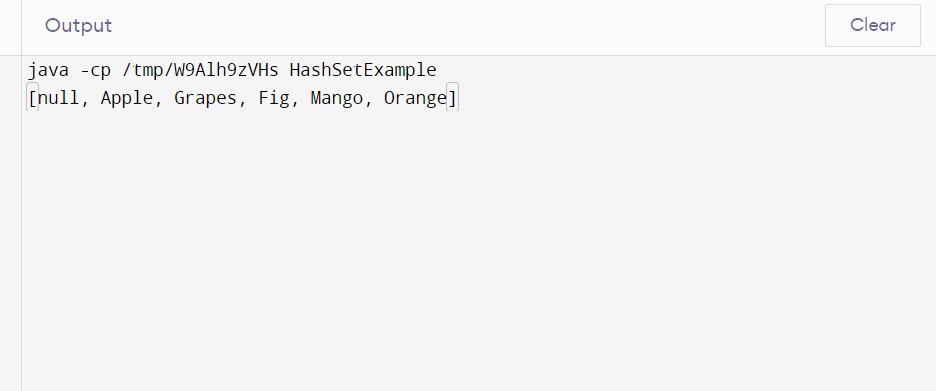
//Displaying HashSet elements System.out.println(hset);

}

}

#### Output:





**Conclusion:**

Hash set and tree set both belong to the collection framework. HashSet is the implementation of the Set interface whereas Tree set implements sorted set.

Tree set is backed by TreeMap while HashSet is backed by a hashmap. HashSet allows a null object.

The tree set does not allow the null object. It throws the null pointer exception.

Hash set use equals method to compare two objects. Tree set use compare method for comparing two objects. Hash set doesn't now allow a heterogeneous object.

Tree set allows a heterogeneous object

#### Sample Viva Questions and Answers:

**Q1. What are the characteristics of Set?**

Ans. A Set is a Collection that cannot contain duplicate elements. It models the mathematical set abstraction. The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited.

#### Q2. Where can you find applications of Set?

Ans. Syntax: Set<T> set = new HashSet<T>(); where T is type of generic. Set<T> set = new LinkedHashSet<T>(); Set<T> set = new TreeSet<T>(); For example: Set<Integer> set = new HashSet<Integer>(); // Creates an empty set of Integer objects.

#### Q3. Difference between HashSet and TreeSet?

Ans. Hash set and tree set both belong to the collection framework. HashSet is the implementation of the Set interface whereas Tree set implements sorted set. Tree set is backed by TreeMap while HashSet is backed by a hashmap.

#### Q4. What data structure does Set uses in background?

Ans. Linked list in Java, uses Doubly Linked List in background to store the data.

#### Q5. Difference between Set and List?

Ans, List is a type of ordered collection that maintains the elements in insertion order while Set is a type of unordered collection so elements are not maintained any order. List allows duplicates while Set doesn't allow duplicate elements.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Roll No.** | **Name**  **Student** | **of** | **Date of**  **Performance** | **Date of**  **Evaluation** | **Grade** | **Sign**  **Student** | **of** | **Sign of**  **Faculty** |
| 0818CS  201108 | Neelesh Singh | |  |  |  |  | |  |

#### EXPERIMENT NO. 3

**Aim / Title:** Remote Method Invocation

**Problem Statement:** Write a program to implement client server based program using Remote Method Invocation

**Objectives:** Implement Remote Method Invocation

**Outcomes:** Student will be able to explain and implement RMI applications

**Pre-requisite:** Basic Knowledge of Java or Linux programming

**Hardware requirements:** Processor: 1 (GHz) RAM: 2 gigabyte (GB) for Hard disk space: 16 GB .Graphics card: DirectX 9 or later with WDDM 1.0 driver.Display: 800x600.

**Software requirements:** Java /Linux

#### Theory:

RMI stands for **Remote Method Invocation**. It is a mechanism that allows an objectresiding in one system (JVM) to access/invoke an object running on another JVM.

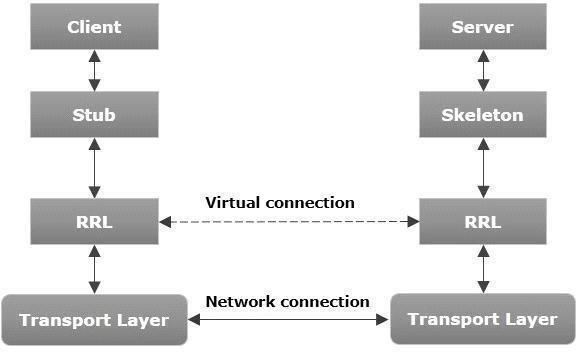
RMI is used to build distributed applications; it provides remote communication between Java programs. It is provided in the package **java.rmi**.

**Architecture of an RMI Application**

In an RMI application, we write two programs, a **server program** (resides on the server) and a **client program** (resides on the client).

* Inside the server program, a remote object is created and reference of that object is made available for the client (using the registry).
* The client program requests the remote objects on the server and tries to invoke its methods.

The following diagram shows the architecture of an RMI application.



Let us now discuss the components of this architecture.

* **Transport Layer** −This layer connects the client and the server. It manages theexisting connection and also sets up new connections.
* **Stub** − A stub is a representation (proxy) of the remote object at client. It resides inthe client system; it acts as a gateway for the client program.
* **Skeleton** − This is the object which resides on the server side. **stub** communicateswith this skeleton to pass request to the remote object.
* **RRL(Remote Reference Layer)** − It is the layer which manages the references madeby the client to the remote object.

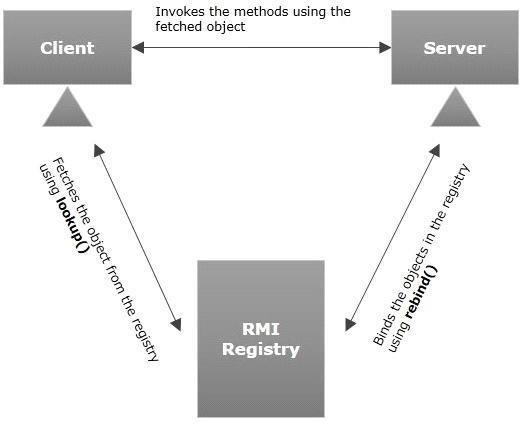
**RMI Registry**

RMI registry is a namespace on which all server objects are placed. Each time the server

creates an object, it registers this object with the RMIregistry (using **bind()** or **reBind()** methods). These are registered using a unique name known as **bind name**.

To invoke a remote object, the client needs a reference of that object. At that time, the client fetches the object from the registry using its bind name (using **lookup()** method).

The following illustration explains the entire process –



**Goals of RMI**

Following are the goals of RMI −

* To minimize the complexity of the application.
* To preserve type safety.
* Distributed garbage collection.
* Minimize the difference between working with local and remote objects.

#### Steps for Creating RMI Program:

* Defining the Remote Interface
* Developing the Implementation Class (Remote Object)
* Developing the Server Program
* Developing the Client Program

#### Program: Hello.java import java.rmi.Remote;

import java.rmi.RemoteException;

// Creating Remote interface for our application public interface Hello extends Remote { void printMsg() throws RemoteException;

}

**ImplExample.java**

#### // Implementing the remote interface

}

}

// Implementing the interface method public void printMsg() {

System.out.println("This is an example RMI program");

public class ImplExample implements Hello {

**Server.java import java.rmi.registry.Registry;**

import java.rmi.registry.LocateRegistry; import java.rmi.RemoteException; import

java.rmi.server.UnicastRemoteObject;

public class Server extends ImplExample { public Server() {}

public static void main(String

args[]) {

try {

// Instantiating the implementation class

ImplExample obj = new ImplExample();

// Exporting the object of implementation class

// (here we are exporting the remote object to the stub)

Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

// Binding the remote object (stub) in the registry

Registry registry = LocateRegistry.getRegistry();

registry.bind("Hello", stub); System.err.println("Server ready");

} catch (Exception e) {

System.err.println("Server exception: " + e.toString()); e.printStackTrace();

}

}

}

#### Client.java

**import java.rmi.registry.LocateRegistry;**

import java.rmi.registry.Registry;

public class Client { private Client() {}

public static void main(String[] args) { try {

// Getting the registry

Registry registry = LocateRegistry.getRegistry(null);

// Looking up the registry for the remote object

Hello stub = (Hello) registry.lookup("Hello");

// Calling the remote method using the obtained object

stub.printMsg();

// System.out.println("Remote method invoked");

} catch (Exception e) {

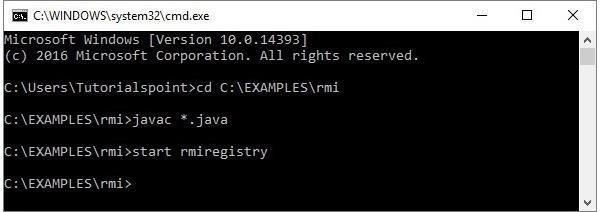
System.err.println("Client exception: " + e.toString()); e.printStackTrace();

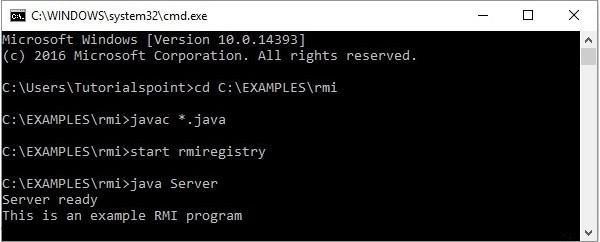
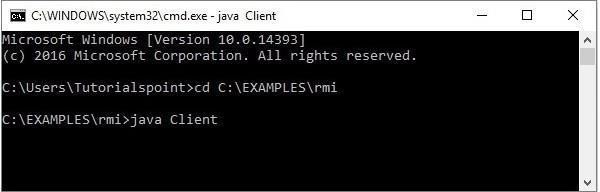
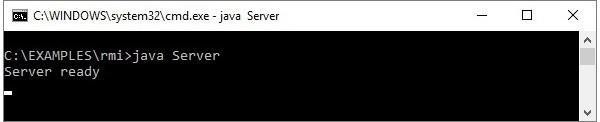
}

}

}

#### Output:





**Sample Viva Questions and Answers:**

#### Q1. What is remote method invocation?

Ans: RMI stands for Remote Method Invocation. It is a mechanism that allows an object residing in one system (JVM) to access/invoke an object running on another JVM. RMI is used to build distributed applications; it provides remote communication between Java programs. It is provided in the package java.

#### Q2. Differentiate RPC and RMI

Ans: RPC vs RMIRPC and RMI are the mechanisms which enable a client to invoke the procedure or method from the server through establishing communication between client and server. The common difference between RPC and RMI is that RPC only supports procedural programming whereas RMI supports object- oriented programming.

#### Q3. What is the role of the java.rmi.Naming Class?

Ans: The java.rmi.Naming class provides methods for storing and obtaining references to remote objects in the remote object registry. Each method of the Naming class takes as one of its arguments a name that is a String in URL format.

#### Q4. What is meant by binding in RMI

Ans: Binding is the process of associating or registering a name for a remote object, which can be used at a later time, in order to look up that remote object. A remote object can be associated with a name using the bind or rebind methods of the Naming class.

#### Q5. What are the steps involved to make work a RMI program

Ans: The following steps must be involved in order for a RMI program to work properly:

⦁ Compilation of all source files.

⦁ Generatation of the stubs using rmic.

⦁ Start the rmiregistry.

⦁ Start the RMIServer.

⦁ Run the client program

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roll**  **No.** | **Name of**  **Student** | **Date of**  **Performance** | **Date of**  **Evaluation** | **Grade** | **Sign of**  **Student** | **Sign of**  **Faculty** |
| 0818CS  201108 | Neelesh Singh |  |  |  |  |  |

## EXPERIMENT NO. 4

**Aim / Title**: Servlet in Java

**Problem Statement**: Develop first Application in Servlet

**Objectives**: Implement and working of Servlet

**Outcomes:** Student will be able to explain and implement servlet

**Pre-requisite**: Basic Knowledge of Java

**Hardware requirements**: Processor: 1 (GHz) RAM: 2 gigabyte (GB) for Hard disk space: 16 GB .Graphics card: DirectX 9 or later with WDDM 1.0 driver.Display: 800x600.

**Software requirements:** Java /Linux

**Theory:**

Java Servlets are programs that run on a Web or Application server and act as a middle layer between requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

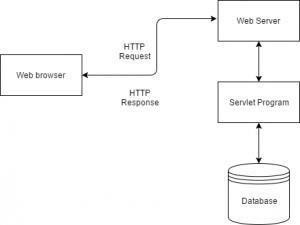
Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

Properties of Servlets :

* Servlets work on the server-side.
* Servlets are capable of handling complex requests obtained from web server.

## Servlets Architecture

The following diagram shows the position of Servlets in a Web Application.



A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

* The servlet is initialized by calling the init() method.
* The servlet calls service() method to process a client's request.
* The servlet is terminated by calling the destroy() method.
* Finally, servlet is garbage collected by the garbage collector of the JVM.

## Execution of Servlets :

Execution of Servlets involves six basic steps:

* 1. The clients send the request to the web server.
  2. The web server receives the request.
  3. The web server passes the request to the corresponding servlet.
  4. The servlet processes the request and generates the response in the form of output.
  5. The servlet sends the response back to the web server.
  6. The web server sends the response back to the client and the client browser displays it on the screen.

**Program: Servlet.java**

***// Import required java libraries import***

***java.io.\*; import javax.servlet.\*; import***

***javax.servlet.http.\*;***

***// Extend HttpServlet class public class HelloWorld***

***extends HttpServlet { private String message;***

***public void init() throws ServletException {***

***// Do required initialization message = "Hello***

***World";***

***}***

***public void doGet(HttpServletRequest request, HttpServletResponse response) throws***

***ServletException, IOException {***

***// Set response content type response.setContentType("text/html");***

***// Actual logic goes here.***

***PrintWriter out = response.getWriter(); out.println("<h1>" +***

***message + "</h1>");***

***}***

***public void destroy() {***

***// do nothing.***

***}***

***}***

**Web.xml**

**<servlet>**

**<servlet-name>HelloWorld</servlet-name>**

**<servlet-class>HelloWorld</servlet-class>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>HelloWorld</servlet-name>**

**<url-pattern>/HelloWorld</url-pattern>**

**</servlet-mapping>**

**Output:**



**Sample Viva Questions and Answers:**

#### Q1. What is servlet?

A servlet is a small program that runs on a server. The term was coined in the context of the Java applet, a small program that is sent as a separate file along with a Web (HTML) page. Java applets, usually intended for running on a client, can result in such services as performing a calculation for a user or positioning an image based on user interaction.

#### Q2. Differentiate servlet and CGI?

1. CGI scripts are written in native OS and stored on the certain directory. On the other hand, servlet programs are usually written in Java which is compiled into Java bytecode and executed in JVM.

2.

CGI is platform specific which makes it hard to switch between operating systems. As against, Servlets can be executed in any operating system which have installed JVM, hence it is platform-independent.

3,

In CGI each arriving client request can generate a separate process while servlet, processes are not created needlessly and share the memory space of the JVM.

4.

CGI scripts are executable programs written in the native OS of the server. In contrast, servlets are compiled to Java bytecode that runs on JVM.

5.

Servlet is more secure than CGI as it uses Java. 6.

The speed, performance and efficiency of the servlet is better than CGI.

CGI scripts can be directly processed. On the contrary, the servlet first translates and compile the program then process it.

7.

When it comes to portability servlet is portable while CGI is not.

#### Q3. What are the main components of Servlet Architecture

1. Client

In this architecture, the web browser acts as a Client. Client or user connected with a web browser

1. Web Server

Web server controls how web user access hosted files, and it’s responsible for processing user request and responses. Here server is software it understands URLs and HTTP protocol.

1. Web Container

A web container is a component in the webserver it interacts with Java servlets. A web container is responsible for managing the lifecycle of servlets, and it also performs the URL mapping task

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roll No.** | **Name of Student** | **Date of**  **Performance** | **Date of Evaluation** | **Grade** | **Sign of Student** | **Sign of Faculty** |
| 0818CS  201108 | Neelesh Singh |  |  |  |  |  |

#### EXPERIMENT NO. 5

**Aim / Title:** Introduction to Servlet

**Problem Statement:** Develop a web application to take values from HTML form and process it on Servlet

**Objectives:** To understand the flow and connection between HTML and JAVA **Outcomes:** Students will be able to build registration and login form application **Prerequisite:** Knowledge of Java

**Hardware requirements:** 4 GB **RAM** minimum, 8 GB **RAM recommended**.

2 GB of available **disk space** minimum, 4 GB **Recommended** (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) 1280 x 800 minimum screen resolution. Microsoft Windows 7/8/10 (32- or 64-bit). The Android Emulator supports 64-bit Windows only.

**Software requirements:** Java /Linux

#### Theory:

The browser uses two methods to pass this information to web server. These methods are GET Method and POST Method.

#### GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? (question mark) symbol as follows

− [http://www.test.com/hello?key1 = value1&key2 = value2](http://www.test.com/hello?key1=value1&key2=value2)

The GET method is the default method to pass information from browser to web server and it produces a long string that appears in your browser's Location:box. Never use the GET method if you have password or other sensitive information to pass to the server. The GET method has size limitation: only 1024 characters can be used in a request string.

This information is passed using QUERY\_STRING header and will be accessible through QUERY\_STRING environment variable and Servlet handles this type of requests using doGet() method.

#### POST Method

A generally more reliable method of passing information to a backend program is the POST method. This packages the information in exactly the same way as GET method, but instead of sending it as a text string after a ? (question mark) in the URL it sends it as a separate message. This message comes to the backend program in the form of the standard input which you can parse and use for your processing. Servlet handles this type of requests using doPost() method.

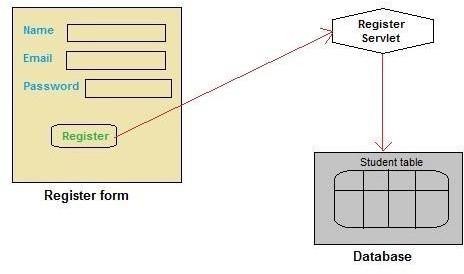
Reading Form Data using Servlet

Servlets handles form data parsing automatically using the following methods depending on the situation −

* getParameter() − You call request.getParameter() method to get the value of a form parameter.
* getParameterValues() − Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
* getParameterNames() − Call this method if you want a complete list of all parameters in the current request.

##### GET Method Example using URL

Here is a simple URL which will pass two values to HelloForm program using GET method. http://localhost:8080/HelloForm?first\_name = ZARA&last\_name = ALI



**Program: LoginServlet.java package net.codejava.servlet;**

**import java.io.IOException; import java.io.PrintWriter;**

**import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;**

**@WebServlet("/loginServlet") public class LoginServlet extends HttpServlet {**

**protected void doPost(HttpServletRequest request,**

**HttpServletResponse response) throws ServletException, IOException {**

**// read form fields**

**String username = request.getParameter("username"); String password = request.getParameter("password");**

**System.out.println("username: " + username); System.out.println("password: " + password);**

**// do some processing here...**

**// get response writer**

**PrintWriter writer = response.getWriter();**

**// build HTML code String htmlRespone = "<html>"; htmlRespone += "<h2>Your username is: " + username + "<br/>"; htmlRespone += "Your password is: " + password + "</h2>"; htmlRespone += "</html>";**

**// return response writer.println(htmlRespone);**





**}**

**}**

#### Index.html

<form name="loginForm" method="post" action="loginServlet"> Username: <input type="text" name="username"/> <br/> Password: <input type="password" name="password"/> <br/>

<input type="submit" value="Login" />

</form>

#### Conclusion:

Java programs are often called applets -- small applications. The applets are distinct from HTML but accessed from within HTML by means of the applet tag, in much the same way that you access a graphic or a sound file or a video file. An applet is called into an HTML document as follows: The hello.

***Sample Viva Questions and Answers:***

#### What is getParameter()method?

Ans. getParameter() method is used to get the parameter values associated with request object of HTML form fields. These fields values are associated to HTTP request after submitting the form. Thismethod returns the String value if the requested parameter is exist or returns null if the requested parameter doesn't exist.

#### Name any 4 tags that come under the form tag?

Ans.

|  |  |
| --- | --- |
| <form> | It defines an HTML form to enter inputs by the used side. |
| <input> | It defines an input control. |
| <textarea> | It defines a multi-line input control. |
| <label> | It defines a label for an input element. |

#### What does web.xml do in a Java web application?

Ans. web. xml defines mappings between URL paths and the servlets that handle requests with those paths. The web server uses this configuration to identify the servlet to handle a given request and call the class method that corresponds to the request method.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Roll No.** | **Name of Student** | **Date of**  **Performance** | **Date of Evaluation** | | **Grade** | **Sign of Student** | **Sign of Faculty** |
| 0818CS  201108 | Neelesh Singh |  |  |  | |  |  |

#### EXPERIMENT NO. 6

**Aim / Title:** Introduction to JDBC

**Problem Statement:** Develop a program to implement the concept of JDBC **Objectives:** To understand the JDBC, different types of connection and database. **Outcomes:** Students will be able to connect there project to database **Prerequisite:** Knowledge of Java

**Hardware requirements:** 4 GB **RAM** minimum, 8 GB **RAM recommended**.

2 GB of available **disk space** minimum, 4 GB **Recommended** (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) 1280 x 800 minimum screen resolution. Microsoft Windows 7/8/10 (32- or 64-bit). The Android Emulator supports 64-bit Windows only.

**Software requirements:** Java /Linux

#### Theory:

JDBC stands for Java Database Connectivity, which is a standard Java API for databaseindependent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

* Making a connection to a database.
* Creating SQL or MySQL statements.
* Executing SQL or MySQL queries in the database.
* Viewing & Modifying the resulting records.

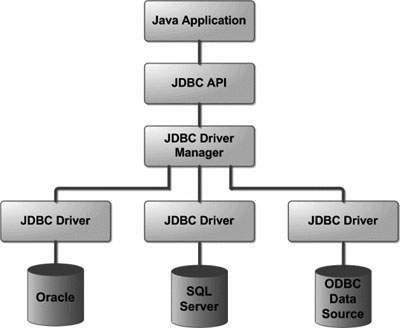
#### JDBC Architecture

The JDBC API supports both two-tier and three-tier processing models for database access but in general, JDBC Architecture consists of two layers −

* JDBC API − This provides the application-to-JDBC Manager connection.
* JDBC Driver API − This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases.

The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application −

#### Common JDBC Components

The JDBC API provides the following interfaces and classes −

* DriverManager − This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.
* Driver − This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.
* Connection − This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
* Statement − You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
* ResultSet − These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
* SQLException − This class handles any errors that occur in a database application.

#### Creating JDBC Application

There are following six steps involved in building a JDBC application −

* **Import the packages** − Requires that you include the packages containing the JDBC classes needed for database programming. Most often, using *import java.sql.\** will suffice.
* **Open a connection** − Requires using the *DriverManager.getConnection()* method to create a Connection object, which represents a physical connection with the database.
* **Execute a query** − Requires using an object of type Statement for building and submitting an SQL statement to the database.
* **Extract data from result set** − Requires that you use the appropriate

*ResultSet.getXXX()* method to retrieve the data from the result set.

* **Clean up the environment** − Requires explicitly closing all database resources versus relying on the JVM's garbage collection.

#### Program: FirstExample.java

import java.sql.\*;

public class FirstExample { static final String DB\_URL = "jdbc:mysql://localhost/TUTORIALSPOINT";

static final String USER = "guest"; static final String PASS = "guest123"; static final String QUERY = "SELECT id, first, last, age FROM Employees";

public static void main(String[] args) {

// Open a connection

try(Connection conn = DriverManager.getConnection(DB\_URL, USER, PASS);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(QUERY);) {

// Extract data from result set while

(rs.next()) {

// Retrieve by column name

System.out.print("ID: " + rs.getInt("id"));

System.out.print(", Age: " + rs.getInt("age"));

System.out.print(", First: " + rs.getString("first"));

System.out.println(", Last: " + rs.getString("last"));

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

#### Output:



**Sample Viva Questions and Answers:**

#### What is JDBC?

Ans. Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access any kind of tabular data, especially relational database. It is part of Java Standard Edition platform, from Oracle Corporation.

#### What is the use of JDBC?

Ans. The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a relational database. JDBC helps you to write Java applications that manage these three programming activities: Connect to a data source, like a database. Send queries and update statements to the database.

1. What are the steps involved in building a Java database application?

Ans. There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

* Register the Driver class
* Create connection
* Create statement
* Execute queries
* Close connection

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roll No.** | **Name of Student** | **Date of**  **Performance** | **Date of Evaluation** | **Grade** | **Sign of Student** | **Sign of Faculty** |
| 0818CS  201108 | Neelesh Singh |  |  |  |  |  |

#### EXPERIMENT NO. 7

**Aim / Title:** Introduction Servlet Cookies

**Problem Statement:** Develop a web application to use Cookies using Servlet

**Objectives:** To understand the the use of Cookies in web application

**Outcomes:** Students will be able apply cookies in web

**Prerequisite:** Knowledge of Java

**Hardware requirements:** 4 GB **RAM** minimum, 8 GB **RAM recommended**.

2 GB of available **disk space** minimum, 4 GB **Recommended** (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) 1280 x 800 minimum screen resolution. Microsoft Windows 7/8/10 (32- or 64-bit). The Android Emulator supports 64-bit Windows only.

**Software requirements:** Java /Linux

#### Theory:

Cookies are text files stored on the client computer and they are kept for various information tracking purpose. Java Servlets transparently supports HTTP cookies.

There are three steps involved in identifying returning users −

* Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
* Browser stores this information on local machine for future use.
* When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

#### The Anatomy of a Cookie

Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser). A servlet that sets a cookie might send headers that look something like this −

HTTP/1.1 200 OK

Date: Fri, 04 Feb 2000 21:03:38 GMT

Server: Apache/1.3.9 (UNIX) PHP/4.0b3

Set-Cookie: name = xyz; expires = Friday, 04-Feb-07 22:03:38 GMT; path = /;

domain = tutorialspoint.com

Connection: close

Content-Type: text/html

#### Setting Cookies with Servlet

Setting cookies with servlet involves three steps −

1. **Creating a Cookie object** − You call the Cookie constructor with a cookie name and a cookie value, both of which are strings.

Cookie cookie = new Cookie("key","value");

Keep in mind, neither the name nor the value should contain white space or any of the following characters

−

[ ] ( ) = , " / ? @ : ;

1. **Setting the maximum ag**e − You use setMaxAge to specify how long (in seconds) the cookie should be valid. Following would set up a cookie for 24 hours.

cookie.setMaxAge(60 \* 60 \* 24);

1. **Sending the Cookie into the HTTP response headers** − You use response.addCookie to add

cookies in the HTTP response header as follows − response.addCookie(cookie);

#### Program:

**Index.html**

**<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="ISO-8859-1">**

**<title>Java Cookies Example</title>**

**</head>**

**<body>**

**<div align="center">**

**<h1>Java Cookies Example</h1>**

**<h2><a href="/add\_cookies">Add Cookies</a></h2>**

**<h2><a href="/read\_cookies">Read Cookies</a></h2>**

**<h2><a href="/delete\_cookies">Delete Cookies</a></h2>**

**</div>**

**</body>**

**</html>**

**AddCookiesServlet.java** *package net.codejava;*

*import java.io.IOException; import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.Cookie; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;*

*@WebServlet("/add\_cookies") public class AddCookiesServlet extends HttpServlet { private static final long serialVersionUID = 1L; private static int cookieCount;*

*public AddCookiesServlet() {*

*}*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*String name = "Cookie" + (++cookieCount);*

*String value = String.valueOf(System.currentTimeMillis()); Cookie cookie = new Cookie(name, value);*

*response.addCookie(cookie);*

*response.getWriter().println("A cookie has been created successfully!");*

*}*

*}*

#### Reading Cookies

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**PrintWriter writer = response.getWriter();**

**Cookie[] cookies = request.getCookies();**

**if (cookies == null) { writer.println("No cookies found");**

**} else {**

**writer.println("Number of cookies: " + cookies.length);**

**for (Cookie aCookie : cookies) { String name = aCookie.getName(); String value = aCookie.getValue();**

**writer.println(name + " = " + value);**

**}**

#### Deleting Cookies

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**PrintWriter writer = response.getWriter();**

**Cookie[] cookies = request.getCookies();**

**if (cookies != null) { for (Cookie aCookie : cookies) { aCookie.setMaxAge(0); response.addCookie(aCookie);**

**}**

**writer.println("All cookies have been deleted!");**

**} else {**

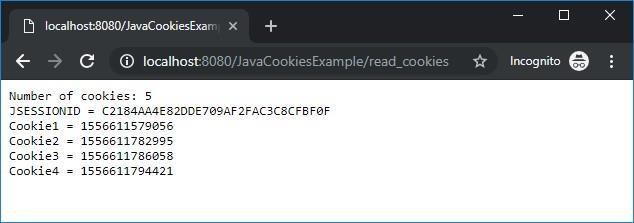
**writer.println("No cookies found");**

**}**

**}**

#### Output:





**Sample Viva Questions and Answers:**

#### What is Cookies?

Ans. Cookies are text files with small pieces of data — like a username and password — that are used to identify your computer as you use a computer network. Specific cookies known as HTTP cookies are used to identify specific users and improve your web browsing experience.

#### What is the use of Cookies?

Ans. A cookie is a small piece of text sent to your browser by a website you visit. It helps the site remember information about your visit, which can make it easier to visit the site again and make the site more useful to you.

#### What are the steps involved in adding cookies in your web app?

Ans.

1. Click 'Tools' (the gear icon) in the browser toolbar.
2. Choose Internet Options.
3. Click the Privacy tab, and then, under Settings, move the slider to the top to block all cookies or to the bottom to allow all cookies, and then click OK.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roll No.** | **Name of Student** | **Date of**  **Performance** | **Date of Evaluation** | **Grade** | **Sign of Student** | **Sign of Faculty** |
| 0818CS  201108 | Neelesh Singh |  |  |  |  |  |

#### EXPERIMENT NO. 8

**Aim / Title:** Introduction Servlet Session

**Problem Statement:** Develop a web application to use Session using Servlet **Objectives:** To understand the the use of Session in web application **Outcomes:** Students will be able apply session in web application **Prerequisite:** Knowledge of Java

**Hardware requirements:** 4 GB **RAM** minimum, 8 GB **RAM recommended**.

2 GB of available **disk space** minimum, 4 GB **Recommended** (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) 1280 x 800 minimum screen resolution. Microsoft Windows 7/8/10 (32- or 64-bit). The Android Emulator supports 64-bit Windows only.

**Software requirements:** Java /Linux

#### Theory:

HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.

Still there are following three ways to maintain session between web client and web server −

#### Cookies

A webserver can assign a unique session ID as a cookie to each web client and for subsequent requests from the client they can be recognized using the recieved cookie.

This may not be an effective way because many time browser does not support a cookie, so I would not recommend to use this procedure to maintain the sessions.

#### Hidden Form Fields

A web server can send a hidden HTML form field along with a unique session ID as follows −

<input type = "hidden" name = "sessionid" value = "12345">

This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data. Each time when web browser sends request back, then session\_id value can be used to keep the track of different web browsers.

This could be an effective way of keeping track of the session but clicking on a regular (<A HREF...>) hypertext link does not result in a form submission, so hidden form fields also cannot support general session tracking.

#### URL Rewriting

You can append some extra data on the end of each URL that identifies the session, and the server can associate that session identifier with data it has stored about that session.

For example, with [http://tutorialspoint.com/file.htm;sessionid](http://tutorialspoint.com/file.htm%3Bsessionid) = 12345, the session identifier is attached as sessionid = 12345 which can be accessed at the web server to identify the client.

URL rewriting is a better way to maintain sessions and it works even when browsers don't support cookies. The drawback of URL re-writing is that you would have to generate every URL dynamically to assign a session ID, even in case of a simple static HTML page.

#### The HttpSession Object

Apart from the above mentioned three ways, servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

You would get HttpSession object by calling the public method getSession() of HttpServletRequest, as below −

HttpSession session = request.getSession();

# Binding Data to a Session

#### Store value in session:

To store a value in a session, use the method setAttribute(key, value) of the HttpSession object. For example, the following statement stores username of the user:

session.setAttribute("username", "Daniel Tran");

Here, the key is username and value is Daniel Tran. Data stored in a session is managed by the server and will be deleted when the session ends.

You can store any kind of object in the session. For example, the following code stores a List of Student objects in the session:

List<Student> students = studentDao.getStudents(); session.setAttribute("listStudent",

students);

# Configure Session Timeout

If a user has been idle (has not made any requests) for a given amount of time, his session expires – which means all the data bound to his session is removed from the server – the session is destroyed. Each server has different default value for global session timeout, e.g. 30 minutes in Apache Tomcat.

You can set session timeout for an individual web application by modifying its web deployment descriptor file (web.xml). For example:

<?xml version="1.0" encoding="UTF-8"?>

<web-app...>

<session-config>

<session-timeout>15</session-timeout>

</session-config>

</web-app>

This overrides the global session timeout set by the server, and sets session timeout to 15 minutes.

You can set timeout value for an individual session programmatically like this: session.setMaxInactiveInterval(300);

#### Program:

**TestSessionServlet.java**

**package net.codejava; import java.io.\*; import java.util.\*; import javax.servlet.\*; import javax.servlet.http.\*; @WebServlet("/test\_session") public class TestSessionServlet extends HttpServlet { private static final long**

**serialVersionUID = 1L; public TestSessionServlet() { super();**

**}**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

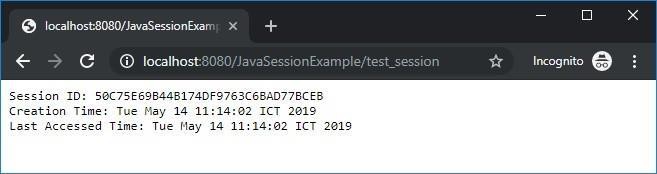
**HttpSession session = request.getSession();**

**PrintWriter writer = response.getWriter(); writer.println("Session ID: " + session.getId()); writer.println("Creation Time: " + new Date(session.getCreationTime())); writer. println("Last Accessed Time: " + new Date(session.getLastAccessedTime()));**

**}**

**}**

#### Output:



1. **What is Session?**

Ans. A session is a group of user interactions with your website that take place within a given time frame. For example a single session can contain multiple page views, events, social interactions, and ecommerce transactions.

#### What is the use of Session?

Ans. Session simply means a particular interval of time.

Session Tracking is a way to maintain state (data) of an user. It is also known as session management in servlet.

Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we needto maintain the state of an user to recognize to particular user

#### What are the steps involved in adding values in Session in your web app?

Ans. To create a new session or to gain access to an existing session, use the HttpServletRequest method getSession(), as shown in the following example:

HttpSession mySession = request.getSession();

getSession() returns the valid session object associated with the request, identified in the session cookie which is encapsulated in the request object. Calling the method with no arguments, creates a session if one does not already exist which is associated with the request. Additionally, calling the method with a Boolean argument creates a session only if the argument is true.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roll No.** | **Name of Student** | **Date of**  **Performance** | **Date of Evaluation** | **Grade** | **Sign of Student** | **Sign of Faculty** |
| 0818CS  201108 | Neelesh Singh |  |  |  |  |  |

#### EXPERIMENT NO.9

**Aim / Title:** Introduction Servlet Filter

**Problem Statement:** Develop a web application using Servlet and use filters **Objectives:** To understand the the use of filters in web application **Outcomes:** Students will be able apply filters in web application **Prerequisite:** Knowledge of Java

**Hardware requirements:** 4 GB **RAM** minimum, 8 GB **RAM recommended**.

2 GB of available **disk space** minimum, 4 GB **Recommended** (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) 1280 x 800 minimum screen resolution. Microsoft Windows 7/8/10 (32- or 64-bit). The Android Emulator supports 64-bit Windows only.

**Software requirements:** Java /Linux

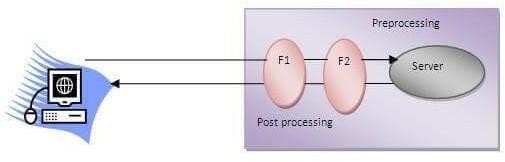
#### Theory:

A **filter** is an object that is invoked at the preprocessing and postprocessing of a request.

It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

The **servlet filter is pluggable**, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.

So maintenance cost will be less.



#### Usage of Filter

* recording all incoming requests
* logs the IP addresses of the computers from which the requests originate
* conversion
* data compression ● encryption and decryption ● input validation etc.

#### Advantage of Filter

1. Filter is pluggable.
2. One filter don't have dependency onto another resource.
3. Less Maintenance

#### Program:

**Index.html**

***<form action="servlet1">***

***Name:<input type="text" name="name"/><br/> Password:<input type="password" name="password"/><br/>***

***<form action="servlet1">***

***Name:<input type="text" name="name"/><br/> Password:<input type="password" name="password"/><br/>***

***<input type="submit" value="login">***

***</form>***

***<input type="submit" value="login">***

***</form>***

**MyFilter.java** *import java.io.IOException; import java.io.PrintWriter; import javax.servlet.\*;*

*public class MyFilter implements Filter{*

*public void init(FilterConfig arg0) throws ServletException {}*

*public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws IOException, ServletException {*

*PrintWriter out=resp.getWriter();*

*String password=req.getParameter("password"); if(password.equals("admin")){ chain.doFilter(req, resp);//sends request to next resource*

*}*

*else{*

*out.print("username or password error!");*

*RequestDispatcher rd=req.getRequestDispatcher("index.html"); rd.include(req, resp);*

*}*

*} public void destroy() {}*

*}*

**AdminServlet.java**

*import java.io.IOException; import java.io.PrintWriter;*

*import javax.servlet.ServletException; import javax.servlet.http.\*;*

*public class AdminServlet extends HttpServlet { public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*response.setContentType("text/html"); PrintWriter out = response.getWriter();*

*out.print("welcome ADMIN"); out.close();*

*}*

*}*

***Web.xml***

*web-app>*

*<servlet>*

*<servlet-name>AdminServlet</servlet-name>*

*<servlet-class>AdminServlet</servlet-class>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>AdminServlet</servlet-name>*

*<url-pattern>/servlet1</url-pattern>*

*</servlet-mapping>*

*<filter>*

*<filter-name>f1</filter-name>*

*<filter-class>MyFilter</filter-class>*

*</filter>*

*<filter-mapping>*

*<filter-name>f1</filter-name>*

*<url-pattern>/servlet1</url-pattern>*

*</filter-mapping>*

*</web-app>*

#### Output:



**Conclusion:**

Java Servlet Filter is the most important web component in java web application. It can intercept both requests and responses in the HTTP web communication. It can extract and manipulate the data exchanged between the client and the web server. So, you can write code to pre-process the request and post-process the response with it. You can do actions such as logging, authentication, manipulate session attributes before the request pass to the called resources. You can also do actions such as encoding, formatting before the response data send back to the client.

#### Sample Viva Questions and Answers:

1. **What are Filters?**

Ans. A filter is an object that is used throughout the pre-and post-processing stages of a request. Filters are mostly used for filtering tasks such as server-side logging, authentication, and authorization, input validation, and so on.

#### What is the use of filter?

Ans. It allows you to declare servlet filters, which are aspects for HTTP requests. A filter chain can intercept an incoming request and/or an outgoing response and modify it as needed. A common example is to have a filter that performs a GZIP compression on a response stream if the user's browser can accept it.

#### What are the steps involved in adding filter in your web app?

Ans. Basically, there are 3 steps to create a filter:

* + Write a Java class that implements the Filter interface and override filter’s life cycle methods.
  + Specify initialization parameters for the filter (optional).
  + Specify filter mapping, either to Java servlets or URL patterns.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roll No.** | **Name of Student** | **Date of Performance** | **Date of Evaluation** | **Grade** | **Sign of Student** | **Sign of Faculty** |
| 0818CS  201108 | Neelesh Singh |  |  |  |  |  |

#### EXPERIMENT NO. 10

**Aim / Title:** Learn Request Dispatcher

**Problem Statement:** Develop a web application and transfer data and move between Servlets

**Objectives:** To understand the inter-servlet communication

**Outcomes:** Students will be able build robust and better web applications

**Prerequisite:** Knowledge of Java

**Hardware requirements:** 4 GB **RAM** minimum, 8 GB **RAM recommended**.

2 GB of available **disk space** minimum, 4 GB **Recommended** (500 MB for IDE + 1.5 GB for Android SDK and emulator system image) 1280 x 800 minimum screen resolution. Microsoft Windows 7/8/10 (32- or 64-bit). The Android Emulator supports 64-bit Windows only.

**Software requirements:** Java /Linux

#### Theory:

The **RequestDispatcher** interface defines an object that receives the request from client and dispatches it to the resource(such as servlet, JSP, HTML file). This interface has following two methods:

**public void forward(ServletRequest request, ServletResponse response)**: It forwards the request from one servlet to another resource (such as servlet, JSP, HTML file).

**public void include(ServletRequest request, ServletResponse response)**: It includes the content of the resource(such as servlet, JSP, HTML file) in the response.

#### Difference between forward() vs include() method

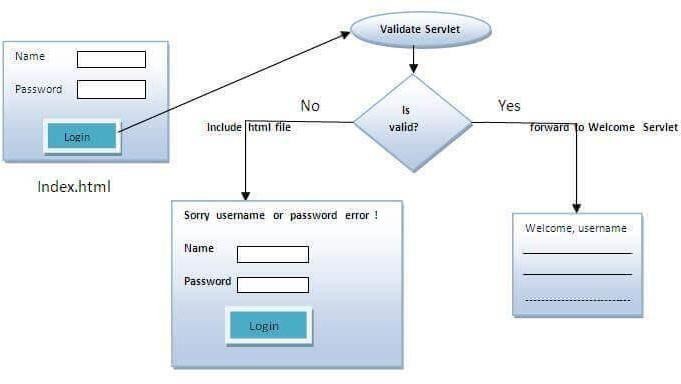
To understand the difference between these two methods, lets take an example: Suppose you have two pages X and Y. In page X you have an include tag, this means that the control will be in the page X till it encounters include tag, after that the control will be transferred to page

Y. At the end of the processing of page Y, the control will return back to the page X starting just after the include tag and remain in X till the end.

#### In this case the final response to the client will be send by page X.

Now, we are taking the same example with forward. We have same pages X and Y. In page X, we have forward tag. In this case the control will be in page X till it encounters forward, after this the control will be transferred to page Y. The main difference here is that the control will not return back to X, it will be in page Y till the end of it.

#### In this case the final response to the client will be send by page Y.



Program:

**Index.html**

*<form action="servlet1" method="post"> Name:<input type="text" name="userName"/><br/>*

*Password:<input type="password" name="userPass"/><br/>*

*<input type="submit" value="login"/>*

*</form>*

**Login.java** *import java.io.\*; import javax.servlet.\*; import javax.servlet.http.\*; public class Login extends HttpServlet {*

*public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*response.setContentType("text/html"); PrintWriter out = response.getWriter();*

*String n=request.getParameter("userName"); String p=request.getParameter("userPass");*

*if(p.equals("servlet"){*

*RequestDispatcher rd=request.getRequestDispatcher("servlet2"); rd.forward(request, response);*

*}*

*else{*

*out.print("Sorry UserName or Password Error!"); RequestDispatcher rd=request.getRequestDispatcher("/index.html");*

*rd.include(request, response);*

*}*

*}*

*}*

**WelcomeServlet.java** *import java.io.\*; import javax.servlet.\*; import*

*javax.servlet.http.\*;*

*public class WelcomeServlet extends HttpServlet {*

*public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*response.setContentType("text/html"); PrintWriter out = response.getWriter();*

*String n=request.getParameter("userName"); out.print("Welcome "+n);*

*}*

*}*

### Web.xml

*<web-app>*

*<servlet>*

*<servlet-name>Login</servlet-name>*

*<servlet-class>Login</servlet-class>*

*</servlet>*

*<servlet>*

*<servlet-name>WelcomeServlet</servlet-name>*

*<servlet-class>WelcomeServlet</servlet-class>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>Login</servlet-name>*

*<url-pattern>/servlet1</url-pattern>*

*</servlet-mapping>*

*<servlet-mapping>*

*<servlet-name>WelcomeServlet</servlet-name>*

*<url-pattern>/servlet2</url-pattern>*

*</servlet-mapping>*

*<welcome-file-list>*

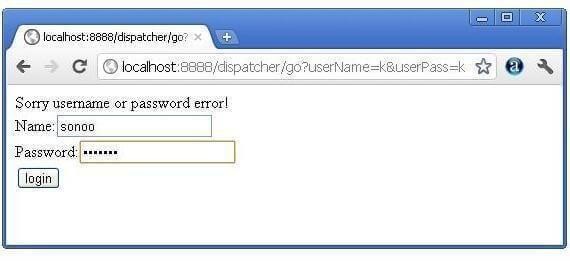
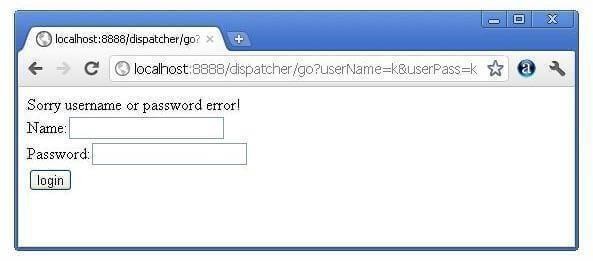
*<welcome-file>index.html</welcome-file>*

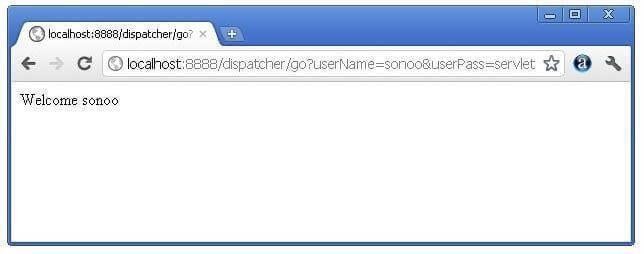
*</welcome-file-list>*

*</web-app>*

#### Output:







**Conclusion:**

HTTP servlets can read HTTP headers and write HTML coding to deliver a response to a browser client. The forward() method is used to transfer the client request to another resource (HTML file, servlet, jsp etc).

When this method is called, the control is transferred to the next resource called. On the other hand, the include() method is used to include the content of the calling file into the called file. Servlets running together in the same server have several ways to communicate with each other.

There are three major reasons to use interservlet communication: Direct servlet manipulation, Servlet reuse, Servlet collaboration.

#### Sample Viva Questions and Answers:

1. **What is a Request Dispatcher?**

Ans. The Request Dispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

#### What is the use of Request Dispatcher?

Ans. The Request Dispatcher interface provides the option of dispatching the client’s request to another web resource, which could be an HTML page, another servlet, JSP etc. It provides the following two methods:

public void forward(Servlet Request request, Servlet Response response)throws Servlet Exception, java.io.IOException

public void include(Servlet Request request, Servlet Response response)throws Servlet Exception, java.io.IOException.

#### What are the steps involved in moving from one servlet to other in your web app?

Ans. To forward request from one servlet to other either you can user Request Dispatcher or Send Redirect. To use Request Dispatcher, you must have to get Servlet Context reference and then you have to call the getRequestDispatcher() method of Servlet Context and using Send Redirect you have to write response. send Redirect("URL").

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Roll No.** | **Name of Student** | **Date of Performance** | **Date of Evaluation** | **Grade** | **Sign of Student** | **Sign of Faculty** |
| 0818CS  201108 | Neelesh Singh |  |  |  |  |  |