# Remote Method Invocation in Java

Difficulty Level : Expert  Last Updated : 29 Oct, 2021

Remote Method Invocation (RMI) is an API that allows an object to invoke a method on an object that exists in another address space, which could be on the same machine or on a remote machine. Through RMI, an object running in a JVM present on a computer (Client-side) can invoke methods on an object present in another JVM (Server-side). RMI creates a public remote server object that enables client and server-side communications through simple method calls on the server object.

**Stub Object:** The stub object on the client machine builds an information block and sends this information to the server. The block consist

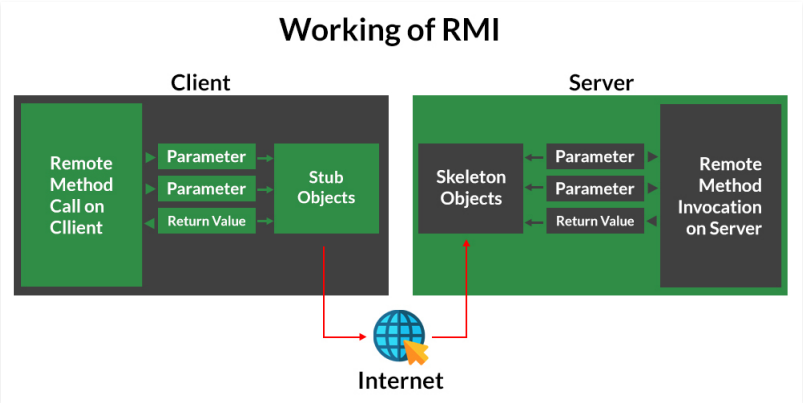
- An identifier of the remote object to be used
- Method name which is to be invoked
- Parameters to the remote JVM

**Skeleton Object:** The skeleton object passes the request from the stub object to the remote object. It performs the following tasks

- It calls the desired method on the real object present on the server.
- It forwards the parameters received from the stub object to the method.

## Working of RMI

The communication between client and server is handled by using two intermediate objects: Stub object (on client side) and Skeleton obje side) as also can be depicted from below media as follows:




**These are the steps to be followed sequentially to implement Interface as defined below as follows:**

1. Defining a remote interface
2. Implementing the remote interface
3. Creating Stub and Skeleton objects from the implementation class using rmic (RMI compiler)
4. Start the rmiregistry
5. Create and execute the server application program
6. Create and execute the client application program.

### Step 1: Defining the remote interface

The first thing to do is to create an interface that will provide the description of the methods that can be invoked by remote clients. This interface should extend the Remote interface and the method prototype within the interface should throw the RemoteException.

**Example:**

```
// Creating a Search interface
import java.rmi.*;
public interface Search extends Remote
{
    // Declaring the method prototype
```

```
    public String query(String search) throws RemoteException;
}
```

**Step 2: Implementing the remote interface**

The next step is to implement the remote interface. To implement the remote interface, the class should extend to UnicastRemoteObject class of java.rmi package. Also, a default constructor needs to be created to throw the java.rmi.RemoteException from its parent constructor in class.

---

```
// Java program to implement the Search interface
import java.rmi.*;
import java.rmi.server.*;
public class SearchQuery extends UnicastRemoteObject
                         implements Search
{
    // Default constructor to throw RemoteException
    // from its parent constructor
    SearchQuery() throws RemoteException
    {
        super();
    }

    // Implementation of the query interface
    public String query(String search)
                        throws RemoteException
    {
        String result;
        if (search.equals("Reflection in Java"))
            result = "Found";
        else
            result = "Not Found";

        return result;
    }
}
```

**Step 3:  Creating Stub and Skeleton objects from the implementation class using rmic**

The rmic tool is used to invoke the rmi compiler that creates the Stub and Skeleton objects. Its prototype is rmic classname. For above program the following command need to be executed at the command prompt
rmic SearchQuery.

**Step 4: Start the rmiregistry**

Start the registry service by issuing the following command at the command prompt start rmiregistry

**Step 5: Create and execute the server application program**

The next step is to create the server application program and execute it on a separate command prompt.

- The server program uses createRegistry method of LocateRegistry class to create rmiregistry within the server JVM with the port number passed as an argument.
- The rebind method of Naming class is used to bind the remote object to the new name.

---

```
// Java program for server application
import java.rmi.*;
import java.rmi.registry.*;
public class SearchServer
{
```

```
    public static void main(String args[])
    {
        try
        {
            // Create an object of the interface
            // implementation class
            Search obj = new SearchQuery();

            // rmiregistry within the server JVM with
            // port number 1900
            LocateRegistry.createRegistry(1900);

            // Binds the remote object by the name
            // geeksforgeeks
            Naming.rebind("rmi://localhost:1900"+
                          "/geeksforgeeks",obj);
        }
        catch(Exception ae)
        {
            System.out.println(ae);
        }
    }
}
```

**Step 6: Create and execute the client application program**

The last step is to create the client application program and execute it on a separate command prompt . The lookup method of the Naming class is used to get the reference of the Stub object.

---

```
// Java program for client application
import java.rmi.*;
public class ClientRequest
{
    public static void main(String args[])
    {
        String answer,value="Reflection in Java";
        try
        {
            // lookup method to find reference of remote object
            Search access =
                (Search)Naming.lookup("rmi://localhost:1900"+
                                      "/geeksforgeeks");
            answer = access.query(value);
            System.out.println("Article on " + value +
                               " " + answer+" at GeeksforGeeks");
        }
        catch(Exception ae)
        {
            System.out.println(ae);
        }
    }
}
```

***Note:*** *The above client and server program is executed on the same machine so localhost is used. In order to access the remote object from another machine, localhost is to be replaced with the IP address where the remote object is present.*

**Important Observations:**

1. RMI is a pure java solution to Remote Procedure Calls (RPC) and is used to create the distributed applications in java.
2. Stub and Skeleton objects are used for communication between the client and server-side.

This article is contributed by **Aakash Ojha**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.