Remote Method Invocation in Java

Nisha Yadav
Student
Department of Information Technology
Maharishi Dayanand University, Gurgaon, Haryana, India

Abstract— Java is a very modern language supporting all major functions that helps users to access the applications easily and reduce a lot of human effort. Remote Method Invocation is used in distributed computation and parallel applications, its integration with Java has proven to be a best option for RMI as Java has provided safe object oriented programming feature has enhanced its performance to top notch. Here this research paper provides an insight the benefits of the RMI's integration with Java, its connectivity with existing legacy systems ,its advantages over traditional RPC model .This paper gives an insight about RMI's nature, behavior with the client and also relationship with JDBC.

Key words: RMI, Java, Object Oriented Programming, RPC Model, JDBC

I. Introduction

RMI allows Java object running on the same or separate computers to communicate with one another via Remote Method calls. In RMI technology the object whose methods make the remote call is called the client object. The compiler running the Java code that calls the remote method is client for the call, and the compiler hosting the object, the process is server for that call.

Distributed computation uses a simple yet direct model known as RMI. The objects that we use in RMI USING Java API can be new Java objects or a wrapper class. This has a tagline Write once run anywhere. RMI is the extension of the Java model because RMI is revolved around Java; it encompasses the strength of safety and also distributed computing in one platform. We will look on to the importance of stubs and marshalling.

II. Client-server relationship

Cooperation of programs in an application can be described by a client server relationship. Client is the one who requests the resources from the server and the server is the one who grant the requests.

RMI method is very similar to RPC which needs both client and server to perform the operation. It is the basic requirement for the functioning of RMI.Stubs and marshalling is an important concept in client- server relationship.

A. *Stubs & MARSHALLING:-*

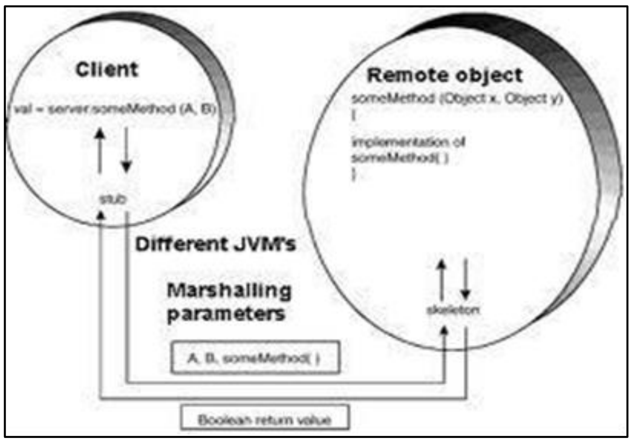

Fig. 1: Marshalling of Parameters

- When client code invokes a remote method on a remote object, it calls on a remote object, it calls an ordinary method on proxy object called as stub.
- The stub resides on client machine and not on the server.
- The stub packages the parameters used on the remote method into a block of bytes.
- This packaging uses a device-independent encoding for each parameter.
- The process of encoding the parameter is called parameter marshalling.
- The purpose of parameters marshalling is converting the parameters into a format suitable for transport from one virtual machine to another.
- The stub method on the client build an information block that consists of an information block that consists of an identifier of the remote object. This is known as the marshalled parameters.
- The stub then sends the information to the server. On the server side , a receiver object performs the following actions for any remote method calls.
  - It unmarshals the parameters.
  - It locates the object to be called.
  - It calls the desired method
  - It captures the marshals the return value or exception of the call.
- The client stub unmarshals the return value or exception from the server. This becomes the return value of the stub call.

III. Advantages/Disadvantages over RPC

RMI is RPC's remote procedure call in Java. It follows object oriented approach, hence it is more efficient than RPC.

RMI 's main focus is to connection to already existing system with the help of native methods. RMI takes a natural and a more direct method in distributed computing.

As RMI is done using Java API this means that it takes more overhead in comparison to RPC.

RPC has to every time convert its arguments between the architecture so that computer could use the corresponding datatype.

The biggest limitation is it can only call methods in Java. Calling of methods written in other languages it has to rely on other technologies like JNI, JDBC etc.,

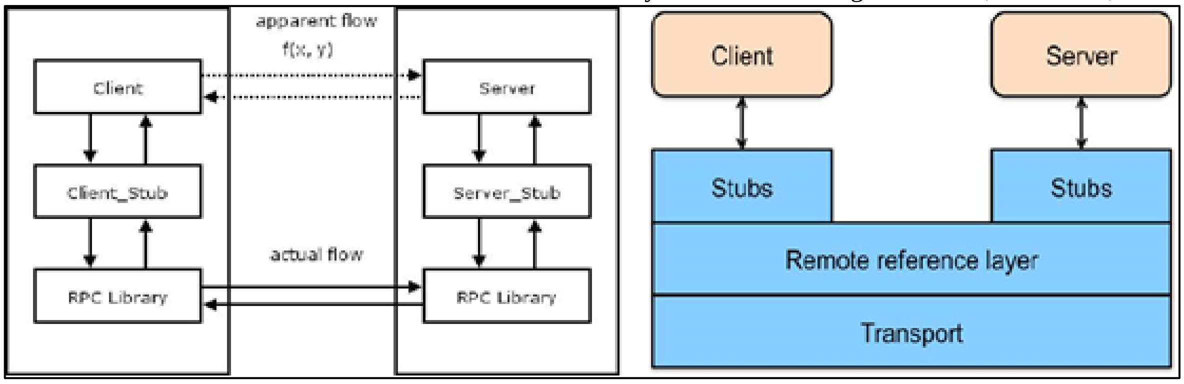

Fig. 2 RPC vs RMI

IV. Relationship with JDBC

Initialization of the OrderServerImpl object by the constructor which in turn creates a connection to a database that has already been specified in a jdbc URL.

After the connection has been created, we use prepareStatement which defines a query that finds all unpaid orders.

Other queries con be defined here for other methods.

The OrderServerImpl server runs not only on the same system as the database, but possibly in the same process.

When a getUnpaid method which is invoked on the RMI server object OrderServerImpl, the precompiled query is executed this returns a JDBC ResultSet object which contains all the matching elements.

Creation of new Order objects for each item in the result set, and putting each into a Vector object (Java's dynamically-sized array) is done. When we finish reading the results then returning of the Vector to the client is done who then displays the results to the user or do whatever else is appropriate.

V. Conclusion

After conducting the detailed analysis of the Remote Method Invocation mechanism in Java, one may conclude that java is one the most secure web development languages and RMI as a feature has only increased its extensibility. Connecting to remote servers has become a more easier task ,it has showed its support in Java but and its connection to JDBC has showed that it can use native methods to bridge existing legacy systems. Even though there are methods similar to RMI, but it has proved its full potential and integration with Java has enhanced its functionality and performance.

Acknowledgement

I would like to thank the entire faculty of Department of IT, Dronacharya College of Engineering, Gurgaon, Haryana for their support, encouragement and patience. Special thanks to Prof. Megha Goel for encouraging me to write this paper.

References

[1] Cay S. Horstmann and Gary Cornell, "Core Java, Volume I--Fundamentals (8th Edition)".
[2] William Grosso ," Java RMI", First Edition October 2001, O'Reilly
[3] http://www.javatpoint.com/RMI
[4] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138781.html
[5] http://en.wikibooks.org/wiki/Java_Programming/Remote_Method_Invocation