Remote Method Invocation in Java

Presenters:
Anil Adhikari
Nabin Bhandari
Shreedhar Acharya

Introduction

- a **Java API** that performs remote method invocation, the object-oriented equivalent of remote procedure calls (RPC).
- Supports direct transfer of serialized Java classes and distributed garbage collection.
- an alternative to low level sockets.
- Instead of creating objects on local machines we create some of the objects on other machines and we communicate with those objects as we would normally do with local objects.

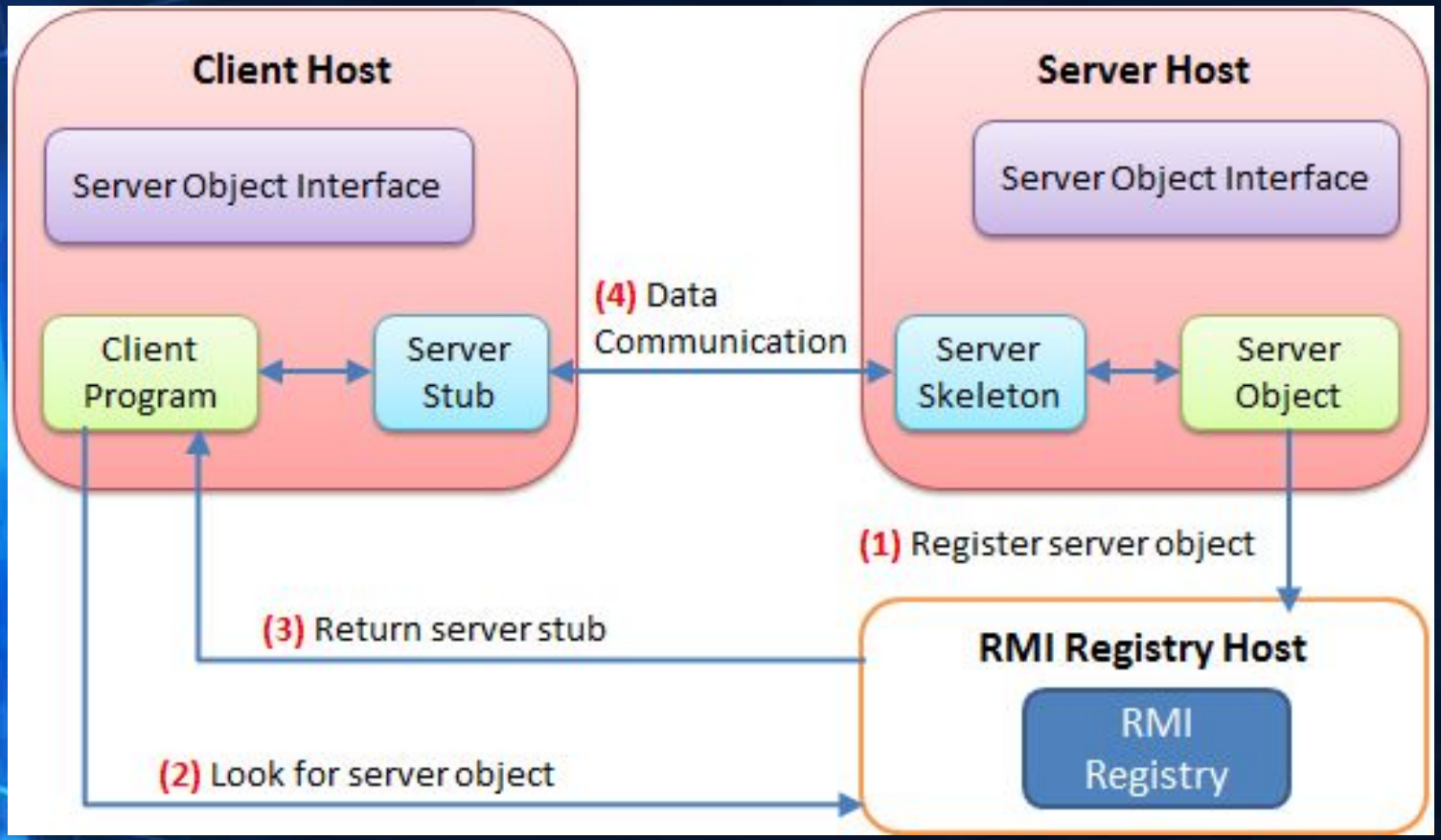
WHAT IS RMI?

- RMI is a core package of the JDK 1.1 and above that can be used to develop distributed application.
- It enables software developers to write distributed applications in which the methods of remote objects can be invoked from other JVMs

Goals of RMI

- Support seamless remote invocations on objects in different java virtual machines.
- Integrate the distributed object model into the Java language in a natural way while retaining most of the Java language's object semantics.
- Make writing reliable distributed applications as simple as possible.
- Preserve the safety provided by the java sun real time environment.

Java RMI Overview



Security

- There are a number of security issues that you should be aware of when developing mission-critical systems in RMI.
- There is no authentication; a client just requests and object (stub), and the server supplies it. Subsequent communication is assumed to b from the same client.
- There is no access control to the objects
- There are no security checks on the RMI registry itself; any caller Is allowed to make request.
- There is no version control between the stubs and the skeletons, making it possible for client to use a down-level stub to access a more recent skeleton breaking release-to-release binary compatibility

Java RMI steps

- ✔ Define the remote interface.
- ✔ Implement the server.
- ✔ Implement the client.
- ✔ Compile the source files.
- ✔ Start the Java RMI registry, Server and Client.

Example

- This example will follow to create a distributed version of the classic Hello World program using Java Remote Method Invocation (Java RMI).
- uses a simple client to make a remote method invocation to a server which may be running on a remote host.
- The files which shall be created are:
  - `Hello.java` - a remote interface.
  - `Server.java` - a remote object implementation that implements the remote interface
  - `Client.java` - a simple client that invokes a method of the remote interface.

Hello.java

```java
//Hello.java
package example.hello;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Hello extends Remote {
    String sayHello() throws RemoteException;
}
```

Server.java

```java
//Server.java
package example.hello;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

public class Server implements Hello {
    public String sayHello() { return "Hello, world!"; }
    public static void main(String args[]) {
        try {
            Server obj = new Server();
            Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

            Registry registry = LocateRegistry.createRegistry(1099);
            registry.bind("Hello", stub);

            System.out.println("Server ready");
        } catch (Exception e) {
            System.err.println("Server exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

Client.java

```java
//Client.java
package example.hello;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Client {
    public static void main(String[] args) {
        String host = (args.length < 1) ? null : args[0];
        try {
            Registry registry = LocateRegistry.getRegistry(host);
            Hello stub = (Hello) registry.lookup("Hello");
            String response = stub.sayHello();
            System.out.println("response: " + response);
        } catch (Exception e) {
            System.err.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

References

- Qusay H. Mahmoud,(1999). Distributed Programming with Java. Greenwich CT: Manning Publications Co.
- https://en.wikipedia.org/wiki/**Java** remote method invocation
- https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/hello/hello-world.html
- http://lycog.com/wp-content/uploads/2011/03/java-rmi-overview.png

Queries?