|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра инструментального и прикладного программного обеспечения |

|  |  |
| --- | --- |
|  | |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2** | |
| **по дисциплине** | |
| **«Архитектура клиент-серверных приложений»** | |
| Выполнил студент группы **ИКБО-01-21** | Маров Г.А. |
| Принял преподаватель кафедры ИиППО | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « » 2023г. |  |
| «Зачтено» | « » 2023г. |  |

Москва

2023

**Теоретическое введение**

Практическая работа №2. Концепция удаленного вызова метода (RMI) в Java.

**Цель**: Знакомство с механизмами удаленного вызова процедур в Java.

RMI - это объектно-ориентированный эквивалент RPC (удаленного вызова процедур). Это называется вызовом удаленного метода. Вызов удаленного метода (RMI) позволяет объекту Java вызывать метод объекта, запущенного на другом компьютере. RMI обеспечивает удаленную связь между Java-программой.

Главной целью разработчиков RMI было предоставление возможности программистам разрабатывать распределенные Java программы, используя такие же синтаксис и семантику, как и при разработке обычных нераспределенных программ. Для этого они должны были преобразовать модель работы классов и объектов в одной виртуальной машине Java (JVM) в новую модель работы классов и объектов в распределенной (несколько JVM) вычислительной среде.

Приложение RMI можно разделить на две части. Одним из них является программой клиентом и другая программа сервер. Сервер программа создает некоторые удаленный объект, сделать их ссылки доступны для клиента для вызова метода на нем. Клиент программа делает запрос для удаленных объектов на сервере и вызвать метод на них. Заглушка и скелет - два важных объекта, используемых для связи с удаленным объектом.

Таким образом, запустив открытый сервер RMI в системе, можно разрешить внешним субъектам взаимодействовать с ним и, возможно, выполнять методы на сервере RMI. Эти методы должны быть определены в реализации Сервера. Как только они вызываются клиентом, они будут выполняться на сервере, а возвращаемые значения будут возвращены клиенту. Еще одна интересная часть заключается в том, что собственный RMI (опять же, я НЕ говорю о JMXRMI) не поддерживает большую часть безопасности, кроме шифрования соединения с использованием SSL.

**Постановка задачи**

Использую информацию из описания данной практической работы, необходимо реализовать удалённый метод решения квадратных уравнений общего вида ax2 + bx + c = 0. При этом, условие уравнения передавать на сервер, а клиентская часть должна получать результат в виде объектов пользовательского класса. Клиент и сервер должны работать на одном хосте.

**Программный код**

Листинг 1 – Класс Client.java

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Client {

public static void main(String[] args) {

try {

Registry registry = LocateRegistry.getRegistry("localhost", 1099);

EquationSolver solver = (EquationSolver) registry.lookup("EquationSolver");

Solution solution = solver.solveEquation(1, -3, 2);

if (solution.hasSolution()) {

System.out.println("Root1: " + solution.getRoot1());

System.out.println("Root2: " + solution.getRoot2());

} else {

System.out.println("The equation has no real solutions.");

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

Листинг 2 – Класс EquationSolver.java

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface EquationSolver extends Remote {

Solution solveEquation(double a, double b, double c) throws RemoteException;

}

Листинг 3 – Класс EquationSolverImpl.java

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

public class EquationSolverImpl extends UnicastRemoteObject implements EquationSolver {

public EquationSolverImpl() throws RemoteException {

super();

}

public Solution solveEquation(double a, double b, double c) throws RemoteException {

double discriminant = b \* b - 4 \* a \* c;

if (discriminant < 0) {

return new Solution(0, 0, false); // No solution

} else if (discriminant == 0) {

double root = -b / (2 \* a);

return new Solution(root, root, true); // One real solution

} else {

double root1 = (-b + Math.sqrt(discriminant)) / (2 \* a);

double root2 = (-b - Math.sqrt(discriminant)) / (2 \* a);

return new Solution(root1, root2, true); // Two real solutions

}

}

}

Листинг 4 – Класс Server.java

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Server {

public static void main(String[] args) {

try {

EquationSolver solver = new EquationSolverImpl();

Registry registry = LocateRegistry.createRegistry(1099);

registry.bind("EquationSolver", solver);

System.out.println("Server is ready");

} catch (Exception e) {

e.printStackTrace();

}

}

}

Листинг 5 – Класс Solution.java

import java.io.Serializable;

public class Solution implements Serializable {

private double root1;

private double root2;

private boolean hasSolution;

public Solution(double root1, double root2, boolean hasSolution) {

this.root1 = root1;

this.root2 = root2;

this.hasSolution = hasSolution;

}

public double getRoot1() {

return root1;

}

public double getRoot2() {

return root2;

}

public boolean hasSolution() {

return hasSolution;

}

}

Для работы RMI необходимо запустить сервер и клиент. Выполним это, предварительно скомпилировав созданные .java файлы 1, 2.



Рисунок 1 – Компиляция .java файлов

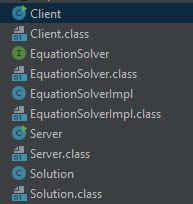


Рисунок 2 – Содержимое файловой директории после компиляции

**Вывод программы**

Теперь мы можем запустить сервер и клиент. Для этого откроем две отдельные консоли и перейдем в папку со скомпилированными файлами. С помощью команды java Server сначала запускаем сервер, а затем введя java Client запускаем клиент. Теперь вся конфигурация собрана, запущена и готова к работе.

Проверим работу системы – в консоли клиента передадим в строковом виде квадратные уравнения для расчета. В ответ от сервера клиент получает корни и печатает их в свой вывод (рисунок 3).

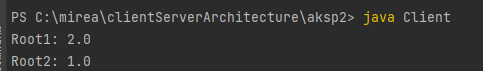


Рисунок 4 – Выполнение запросов со стороны клиента к серверу и успешное получение результата вычислений

**Вывод**

В результате выполнения данной практической работы было проведено ознакомление с механизмами удаленного вызова процедур в Java.