Invocação de Métodos Remotos (RMI)
Conversão de Moedas

Carlos Antônio de Oliveira Neto
Pedro Henrique Bezerra Cavalcante

August 2018

Sucintamente, o projeto consiste na implementação de um projeto cliente/servidor onde o cliente realiza processamento de dados em um servidor que está remotamente localizado, via rede.

Para esse projeto, especificamente, se trata de um conversor de moedas, onde o cliente pode fazer conversões de uma moeda para outra ou, até mesmo, de uma moeda para todas as disponíveis na aplicação.

O Projeto contem 3 classes java e uma interface

Em um primeiro momento, definiu-se a interface que será implementada pela classe de conversão das moedas:

```
1  public interface IConvert extends Remote{
2      /**
3       * Converte um dado valor de uma moeda A para uma moeda
            B.
4       * @param value - valor a ser convertido
5       * @param from - moeda de origem
6       * @param to - moeda de destino
7       * @return Valor convertido
8       * @throws RemoteException
9       */
10     public Double currencyAToB(Double value, String from,
            String to) throws RemoteException;
11     /**
12      * Converte o valor de uma moeda A para todas as 20
            moedas da base de dados.
13      * @param value - valor a ser convertido
14      * @param from - moeda de origem
15      * @return Lista com valores convertidos
16      * @throws RemoteException
17      */
18     public List<Double> currencyAToAll(Double value, String
            from) throws RemoteException;
19
20 }
```

Temos, nessa classe, a definição dos dois métodos propostos: o `currencyAToB` que recebe um valor, representando a quantidade de moeda, o `from` que indica a moeda de origem e o `to`, representando a moeda de destino.

Para nossa aplicação, foi utilizada uma API externa que retorna (na versão free) um JSON com os valores da moeda em conversão para USD (Dólar Americano). Veja a implementação:

```
1  @SuppressWarnings("serial")
2  public class Converter extends UnicastRemoteObject
       implements IConvert {
3      //Atributos utilizados no acesso da API
4      public static final String ACCESS_KEY = "<
           put_your_access_key_here>";
5      public static final String BASE_URL = "http://apilayer.
           net/api/";
6      public static final String ENDPOINT = "live";
7      public static final String STATIC_DOLAR = "USD";
8      static CloseableHttpClient httpClient = HttpClients.
           createDefault();
9
10     /**
11      * Construtor padrao
12      * @throws RemoteException
13      */
14     protected Converter() throws RemoteException {
15         super();
16     }
17
18 /**
19      * Converte um dado valor de uma moeda A para uma moeda
            B.
20      * @param value - valor a ser convertido
21      * @param from - moeda de origem
22      * @param to - moeda de destino
23      * @return Valor convertido
24      * @throws RemoteException
25      */
26     @Override
27     public Double currencyAToB(Double value, String from,
           String to) throws RemoteException {
28         Double result = sendLiveRequest(from, to);
29         if (result == null) {
30             System.exit(0);
31         }
32         return value * result;
33     }
34
35     /**
36      * Converte o valor de uma moeda A para todas as 32
            moedas da base de dados.
37      * @param value - valor a ser convertido
38      * @param from - moeda de origem
39      * @return Lista com valores convertidos
40      * @throws RemoteException
41      */
42
43     @Override
```

```
44      public List<Double> currencyAToAll(Double value, String
            from) throws RemoteException {
45          List<Double> result = new ArrayList<Double>();
46          result.add(value * sendLiveRequest(from, "DKK"));
47          result.add(value * sendLiveRequest(from, "NOK"));
48          result.add(value * sendLiveRequest(from, "SEK"));
49          result.add(value * sendLiveRequest(from, "CZK"));
50          result.add(value * sendLiveRequest(from, "GBP"));
51          result.add(value * sendLiveRequest(from, "TRY"));
52          result.add(value * sendLiveRequest(from, "INR"));
53          result.add(value * sendLiveRequest(from, "IDR"));
54          result.add(value * sendLiveRequest(from, "PKR"));
55          result.add(value * sendLiveRequest(from, "THB"));
56          result.add(value * sendLiveRequest(from, "USD"));
57          result.add(value * sendLiveRequest(from, "AUD"));
58          result.add(value * sendLiveRequest(from, "CAD"));
59          result.add(value * sendLiveRequest(from, "SGD"));
60          result.add(value * sendLiveRequest(from, "HKD"));
61          result.add(value * sendLiveRequest(from, "TWD"));
62          result.add(value * sendLiveRequest(from, "NZD"));
63          result.add(value * sendLiveRequest(from, "EUR"));
64          result.add(value * sendLiveRequest(from, "HUF"));
65          result.add(value * sendLiveRequest(from, "CHF"));
66          result.add(value * sendLiveRequest(from, "JPY"));
67          result.add(value * sendLiveRequest(from, "ILS"));
68          result.add(value * sendLiveRequest(from, "CLP"));
69          result.add(value * sendLiveRequest(from, "PHP"));
70          result.add(value * sendLiveRequest(from, "MXN"));
71          result.add(value * sendLiveRequest(from, "ZAR"));
72          result.add(value * sendLiveRequest(from, "BRL"));
73          result.add(value * sendLiveRequest(from, "MYR"));
74          result.add(value * sendLiveRequest(from, "RUB"));
75          result.add(value * sendLiveRequest(from, "KRW"));
76          result.add(value * sendLiveRequest(from, "CNY"));
77          result.add(value * sendLiveRequest(from, "PLN"));
78
79          return result;
80      }
81
82       /**
83       * Metodo auxiliar criado para extrair da API o valor de
                1 unidade da moeda de origem e da de destino
84       * em dolares, para entao ser feita a conversao de uma
              moeda para a outra.
85       * @param value - valor a ser convertido
86       * @param from - moeda de origem
87       * @return o valor convertido de uma moeda para outra ou
                NULL no caso de requisicoes com erro da API
88       */
89      private Double sendLiveRequest(String from, String to) {
90
91          //Variaveis auxiliares
92          Double from2 = 1.0;
93          Double to2 = 1.0;
```

```
94
95          // Inicializa o objeto HttpGet com a URL para mandar
                a requisicao para a API
96          HttpGet get = new HttpGet(BASE_URL + ENDPOINT + "?
                access_key=" + ACCESS_KEY);
97          try {
98              CloseableHttpResponse response = httpClient.
                    execute(get);
99              HttpEntity entity = response.getEntity();
100
101         // Converte a resposta JSON em um objeto equivalente
                em Java
102             JSONObject exchangeRates = new JSONObject(
                    EntityUtils.toString(entity));
103
104             //Lanca mensagem no console de que o acesso a
                    API foi iniciado
105             System.out.println("Live Currency Exchange Rates
                    ");
106
107             //Variavel utilizada para capturar se a
                    requisicao a API teve status ou nao
108             boolean status = exchangeRates.getBoolean("
                    success");
109
110             if (!sucesso) {
111
112                 String codigoErro = exchangeRates.
                        getJSONObject("error").getString("code");
113
114                 String infoErro = exchangeRates.
                        getJSONObject("error").getString("info");
115
116                 System.out.println("API reached its peak of
                        access.");
117                 System.out.println("Error: " + codigoErro);
118                 System.out.println("Message: " + infoErro);
119                 System.exit(0);
120             }
121             //Valor equivalente a 1 dolar na moeda de origem
122             System.out.println("Converting " + STATIC_DOLAR
                    + " in " + from + ": "
123                     + exchangeRates.getJSONObject("quotes").
                            getDouble(STATIC_DOLAR + from));
124
125             //Divisao para transformar o valor recuperado da
                     API em 1 unidade da moeda de origem
126             from2 = 1 / exchangeRates.getJSONObject("quotes"
                    ).getDouble(STATIC_DOLAR + from);
127
128             //Impressao no console
129             System.out.println("1 unity of " + from + " in
                    USD: " + from2);
130
```

```
131             //Valor equivalente a 1 dolar na moeda de
                    destino
132             System.out.println("Converting " + STATIC_DOLAR
                    + " in " + to + ": "
133                     + exchangeRates.getJSONObject("quotes").
                            getDouble(STATIC_DOLAR + to));
134
135             //Divisao para transformar o valor recuperado da
                     API em 1 unidade da moeda de destino
136             to2 = 1 / exchangeRates.getJSONObject("quotes").
                    getDouble(STATIC_DOLAR + to);
137
138             //Impressao no console
139             System.out.println("1 unity of " + to + " in USD
                    : " + to2);
140
141             //Impressao no console do resultado da conversao
                     da moeda de origem para a de destino
142             System.out.println(from + " IN " + to + ": " +
                    from2 / to2);
143
144             response.close();
145             return from2 / to2;
146
147
148         } catch (ClientProtocolException e) {
149             e.printStackTrace();
150         } catch (IOException e) {
151             e.printStackTrace();
152         } catch (ParseException e) {
153             e.printStackTrace();
154         } catch (JSONException e) {
155             e.printStackTrace();
156         }
157         return null;
158
159     }
160 }
```

Em sequência, temos a implementação do nosso Cliente devidamente falando, onde será a parte executada que fará as requisições para o servidor.

```
1  public class Client {
2      public static void main(String[] args) throws
           MalformedURLException, RemoteException,
           NotBoundException {
3          // Referencia de objeto para o stub do servidor.
               Usado para chamada de metodos remotos.
4          IConvert stub = (IConvert) Naming.lookup("rmi://
               localhost/CurrencyConverter");
5
6          //Moedas de origem para conversao
7          String[] values = { "DKK - Coroa Dinamarquesa",
8          "NOK - Coroa Norueguesa",
9          "SEK - Coroa Sueca",
```

```
10          "CZK - Coroa Tcheca",
11          "GBP - Libra Esterlina",
12          "TRY - Lira Turca",
13          "INR - Rupia Indiana",
14          "IDR - Rupia Indonesia",
15          "PKR - Rupia Paquistanesa",
16          "THB - Baht Tailandes",
17          "USD - Dolar Americano",
18          "AUD - Dolar Australiano",
19          "CAD - Dolar Canadense",
20          "SGD - Dolar de Cingapura",
21          "HKD - Dolar de Hong Kong",
22          "TWD - Dolar de Taiwan",
23          "NZD - Dolar Neozelandes",
24          "EUR - Euro",
25          "HUF - Forint Hungaro",
26          "CHF - Franco Suico",
27          "JPY - Iene Japones",
28          "ILS - Novo Shekel Israelense",
29          "CLP - Peso Chileno",
30          "PHP - Peso Filipino",
31          "MXN - Peso Mexicano",
32          "ZAR - Rand Sul-africano",
33          "BRL - Real Brasileiro",
34          "MYR - Ringgit Malaio",
35          "RUB - Rublo Russo",
36          "KRW - Won Sul-coreano",
37          "CNY - Yuan Renminbi Chines",
38          "PLN - Zloty Polones" };
39
40          //Moedas de destino da conversao. NOTA: Ao deixar na
                opcao "SELECIONE", a moeda de origem sera
               convertida para todas as outras.
41          String[] values2 = { "SELECIONE",
42          "DKK - Coroa Dinamarquesa",
43          "NOK - Coroa Norueguesa",
44          "SEK - Coroa Sueca",
45          "CZK - Coroa Tcheca",
46          "GBP - Libra Esterlina",
47          "TRY - Lira Turca",
48          "INR - Rupia Indiana",
49          "IDR - Rupia Indonesia",
50          "PKR - Rupia Paquistanesa",
51          "THB - Baht Tailandes",
52          "USD - Dolar Americano",
53          "AUD - Dolar Australiano",
54          "CAD - Dolar Canadense",
55          "SGD - Dolar de Cingapura",
56          "HKD - Dolar de Hong Kong",
57          "TWD - Dolar de Taiwan",
58          "NZD - Dolar Neozelandes",
59          "EUR - Euro",
60          "HUF - Forint Hungaro",
61          "CHF - Franco Suico",
```

```
62          "JPY - Iene Japones",
63          "ILS - Novo Shekel Israelense",
64          "CLP - Peso Chileno",
65          "PHP - Peso Filipino",
66          "MXN - Peso Mexicano",
67          "ZAR - Rand Sul-africano",
68          "BRL - Real Brasileiro",
69          "MYR - Ringgit Malaio",
70          "RUB - Rublo Russo",
71          "KRW - Won Sul-coreano",
72          "CNY - Yuan Renminbi Chines",
73          "PLN - Zloty Polones" };
74
75          Double value = 0.0;
76
77          //Valida o valor a ser utilizado na conversao.
78          while (value <= 0.0) {
79              try {
80                  value = Double.parseDouble(JOptionPane.
                        showInputDialog("Insert a value:"));
81              }catch(NullPointerException ne) {
82                  System.exit(0);
83              }catch(NumberFormatException nfe) {
84                  JOptionPane.showMessageDialog(null, "Wrong
                        format. Please insert a  valid number.");
85              }
86          }
87
88          //Tela de selecao da moeda de origem
89          Object objetoFrom = JOptionPane.showInputDialog(null
                , "Convert Source", "CurrencyConverter",
90                  JOptionPane.DEFAULT_OPTION, null, values, "
                        DKK");
91          String selectedFrom = null;
92          String selectedTo = null;
93          if (objetoFrom != null) {
94              selectedFrom = objetoFrom.toString().substring
                    (0, 3);
95              System.out.println(selectedFrom);
96              System.out.println(value);
97          } else {
98              System.out.println("User cancelled.");
99              System.exit(0);
100         }
101
102     //Tela de selecao da moeda de destino (ou para o caso de
            nao selecionar nenhuma)
103         Object objetoTo = JOptionPane.showInputDialog(null,
                "Convert Source", "CurrencyConverter",
104                 JOptionPane.DEFAULT_OPTION, null, values2, "
                        SELECIONE");
105
106         //Lista auxiliar utilizada para receber o retorno da
                conversao de uma moeda para todas
```

```
107         List<Double> result = new ArrayList<Double>();
108         if (objetoTo != null && !objetoTo.equals("SELECIONE"
                ")) {
109             selectedTo = objetoTo.toString().substring(0, 3)
                    ;
110             Double resultado = 0.0;
111             try{
112             //Chamada do metodo remoto de conversao de uma
                    moeda para outra
113                 resultado = stub.currencyAToB(value,
                        selectedFrom, selectedTo);
114             } catch (UnmarshalException e) {
115                 System.out.println("Oops, I did it again.
                        The API reached its limit and is tired.
                        Contact the developer group to get a new
                        access key.");
116                 System.exit(0);
117             }
118
119             System.out.println(selectedFrom + " to "+
                    selectedTo + ": " + resultado);
120         } else if (objetoTo.equals("SELECIONE")) {
121
122             try{
123             //Chamada do metodo remoto de conversao de uma
                    moeda para todas as outras
124                 result = stub.currencyAToAll(value,
                        selectedFrom);
125             } catch (UnmarshalException e) {
126                 System.out.println("Oops, I did it again.
                        The API reached its limit and is tired.
                        Contact the developer group to get a new
                        access key.");
127                 System.exit(0);
128             }
129             for (int i = 0; i < result.size(); i++) {
130                 System.out.println(values[i] + ": " + result
                        .get(i));
131             }
132         } else {
133             System.out.println("User cancelled.");
134             System.exit(0);
135         }
136
137
138     }
139 }
```

Foi utilizado um pequeno JOptionPane onde o usuário seleciona as moedas a serem convertidas (ou deixa em SELECIONE para converter em todas as moedas).

A lógica se baseia no fato de que a API devolve tudo convertido para Dólar Americano. Sendo assim, pode-se aplicar uma divisão para que se saiba o valor na moeda de destino e multiplicar pelo valor requisitado.

Por último, temos o arquivo do servidor que consiste na criação do registro juntamente com a porta em que vai ocorrer a comunicação entre o cliente e o servidor. Depois, registra a url de execução do servidor (no caso, localhost)

```
/**
 * Classe criada para rodar o servidor remoto.
 * @author carlosant
 * @author pedrohcavalcante
 */
public class Server {

    /**
     * Metodo principal.
     * @param args
     * @throws RemoteException
     * @throws MalformedURLException
     */
    public static void main (String [] args) throws
        RemoteException, MalformedURLException {
        //Execucao do modulo de referencia remota
        LocateRegistry.createRegistry(1099);

        //Instanciacao do objeto
        Converter converter = new Converter();

        //Registro no modulo de referencia remota
        Naming.rebind("rmi://localhost/CurrencyConverter",
            converter);

        //Mensagem impressao em console para sinalizar que o
             servidor esta funcionando
        System.out.println("Running!");
    }
}
```

Para se executar a aplicação, deve-se, primeiro, executar o arquivo Server.java, iniciando assim o servidor da aplicação. Em seguida, executa o Client.java.

Vale ressaltar que para o sucesso da execução, é necessário o uso de alguns arquivos .jar, sendo eles:

- commons-logging-1.1.2.jar
- httpclient-4.5.6.jar
- httpcore-4.4.10.jar
- json-20180813.jar

Caso algum desses .jar esteja faltando no classpath do projeto, não será possível proceder com a compilação e execução.

Veja, a seguir, algumas telas que mostram a execução do projeto:

Execução 1: convertendo Real Brasileiro (BRL) para Dólar Americano (BRL)

Primeiramente, deve-se inserir a quantidade de moeda que o usuário deseja converter.

Em seguida, seleciona a moeda de origem (no caso, BRL)

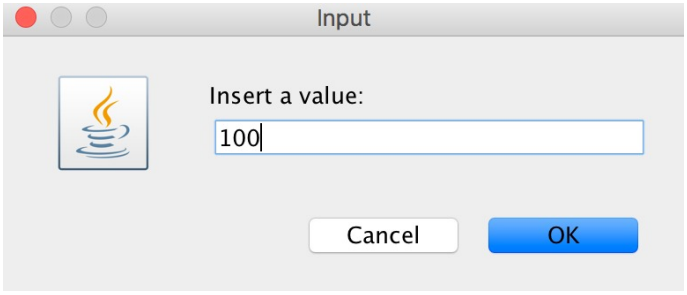

Figure 1: Inserindo valor quantitativo de moeda a ser convertido

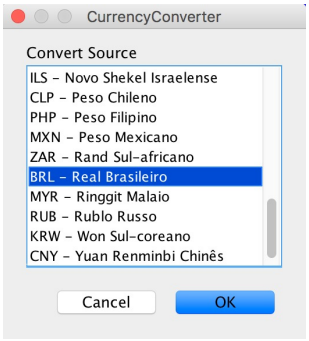

Figure 2: Selecionando a moeda de origem

Então, seleciona-se a moeda de destino (no caso, USD)

Figure 3: Selecionando a moeda de destino

Por fim, tem-se o resultado:

```
BRL
100.0
BRL to USD: 24.346882698179975
```

Figure 4: Resultado da conversão de BRL para USD

Execução 2: convertendo Real Brasileiro (BRL) para todas as moedas disponíveis na aplicação:

Primeiramente, deve-se inserir a quantidade de moeda que o usuário deseja converter.

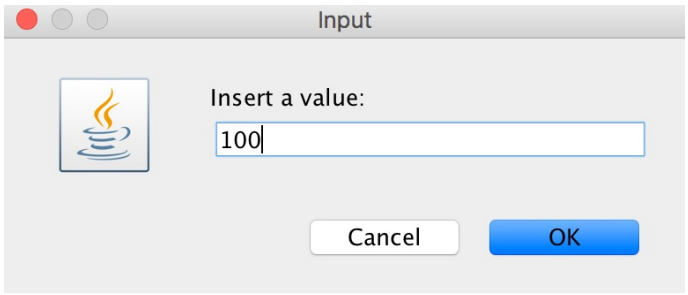

Figure 5: Inserindo valor quantitativo de moeda a ser convertido

Em seguida, seleciona a moeda de origem (no caso, BRL)

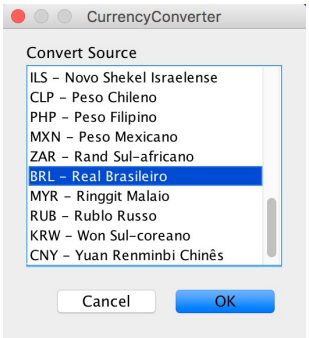

Figure 6: Selecionando a moeda de origem

Neste caso, deve-se deixar a seleção no valor default, SELECIONE:

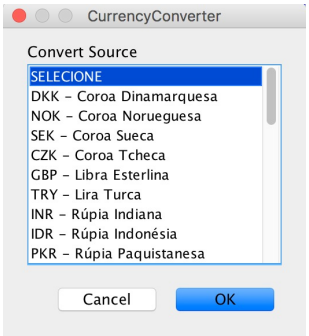

Figure 7: Selecionando a “moeda” Selecione

Por fim, tem-se o resultado:

Figure 8: Resultado da conversão de BRL para todas as moedas disponíveis