Universidade do Minho
Escola de Engenharia

Ferramenta de Monitorização

Gestão de Redes : Trabalho Prático 2

Mestrado Integrado em Engenharia Informática
Universidade do Minho

1º Semestre
2017-2018

Gonçalo Pereira

a74413

10 de fevereiro de 2018

Contents

1 Resolução

1.1 Questão I

Para elaboração da ferramenta de monitorização da ocupação percentual das partições de um Sistema Operativo foi feito um módulo em JAVA, com recurso a uma API pré-compilada e *open source*, de nome **SNMP4J**. Esta API em questão apresenta vários métodos que permitem a **ligação do agente** ao cliente SNMP, devolvendo os resultados da execução dos comandos pedidos.

Inicialmente, terá sido feita a inclusão dos seguintes métodos, já implementados, para iniciar uma sessão SNMP, lidar com os vários tipos de dados e forma de se lidar com os mesmos.

```java
/**
* Start the Snmp session.
*/
private void begin() throws IOException {
    TransportMapping transport = new DefaultUdpTransportMapping();
    snmp = new Snmp(transport);
    // Do not forget this line!
    transport.listen();
}

/**
* Method which takes a single OID and returns the response from the agent as a String.
* @param oid
* @return
* @throws IOException
*/
public String getAsString(OID oid) throws IOException {
    ResponseEvent event = get(new OID[] { oid });
    return event.getResponse().get(0).getVariable().toString();
}

/**
* This method is capable of handling multiple OIDs
* @param oids
* @return
* @throws IOException
*/
public ResponseEvent get(OID oids[]) throws IOException {
    PDU pdu = new PDU();
    for (OID oid : oids) {
        pdu.add(new VariableBinding(oid));
    }
    pdu.setType(PDU.GET);
    ResponseEvent event = snmp.send(pdu, getTarget(), null);

    if(event != null) {
        return event;
    }
        throw new RuntimeException("GET timed out");
}

/**
* This method returns a Target, which contains information about
* where the data should be fetched and how.
* @return
*/
private Target getTarget() {
    Address targetAddress = GenericAddress.parse(address);
    CommunityTarget target = new CommunityTarget();
    target.setCommunity(new OctetString("public"));
    target.setAddress(targetAddress);
    target.setRetries(2);
    target.setTimeout(1500);
    target.setVersion(SnmpConstants.version2c);
return target;
}
```

Figure 1: Métodos SNMP4J.

De seguida, e como teríamos que lidar com múltiplos dados, foram implementadas as seguintes funções para se lidar com percentagens e conversões para *gigabytes*, sem esquecendo a função para truncar os valores a duas casas deci-

mais, preferindo uma versão optimizada extraída através deste link. Por fim, foi também criada uma função para se validar o `IP` que seria recebido do input do utilizador.

```
public static double round(double value, int places) {
    if (places < 0)
        throw new IllegalArgumentException();

    BigDecimal bd = new BigDecimal(value);
    bd = bd.setScale(places, RoundingMode.HALF_UP);
    return bd.doubleValue();
}

public static boolean validateIP (String ip) {
    try {
        if ( ip == null || ip.isEmpty() ) {
            return false;
        }

        String[] parts = ip.split( "\\." );
        if ( parts.length != 4 ) {
            return false;
        }

        for ( String s : parts ) {
            int i = Integer.parseInt( s );
            if ( (i < 0) || (i > 255) ) {
                return false;
            }
        }
        return !ip.endsWith(".");
    }
    catch (NumberFormatException nfe) {
        return false;
    }
}

public static String parsePercentage(String value1, String value2){
    double totalSize = (Double.parseDouble(value1) * 4096) / 100000000;
    double totalUsed = (Double.parseDouble(value2) * 4096) / 100000000;
    String parsed;
    if(totalSize > 0) parsed = String.valueOf(round(((totalSize-totalUsed)/totalSize) * 100, 2));
    else parsed = "0";
return parsed;
}

public static String parseGigabytes(String value){
    double totalGigabytes = Double.parseDouble(value);
    String parsed = String.valueOf(round((totalGigabytes*4096)/1000000000, 2));
    return parsed;
}
```

Figure 2: Métodos implementados para *parsing* dos dados.

Aquando da execução do programa, o método `main` estará encarregue de receber uma `string` para dar *parse* para se inicializar o agente SNMP (no IP fornecido), atualizando, de seguida, os valores extraídos num *array* bi-dimensional de `Object`.

```java
public static void main(String[] args) throws IOException {
    Scanner sc = new Scanner(System.in);
    System.out.println("Input the desired IP:\n");
    String st = sc.nextLine();

    AgentSNMP agent = new AgentSNMP("udp:" + st + "/161");
    agent.begin();
    while(true){
        Object[][] values = updateInfo(agent);
        createHTML(values);
        try{
            Thread.sleep(10000);
        }catch(InterruptedException e){

        }
    }
}

public static Object[][] updateInfo(AgentSNMP agent) throws IOException{
    Object[][] strings;
    globalOID = "1.3.6.1.2.1.25.2.3.1";
    int i = 0;
    int j = 0;
    String test;

    while(!(agent.getAsString(new OID(globalOID + ".3." + (i+1))).equals("Null"))){
        j++;
        i++;
    }

    strings = new Object[i][j];

    for(i = 0; i < j; i++){
        for(int k = 0; k < 3; k++){
            if(k == 0) strings[i][k] = agent.getAsString(new OID(globalOID + ".3." + (i+1)));
            if(k == 1) strings[i][k] = parseGigabytes(agent.getAsString(new OID(globalOID + ".5." + (i+1))));
            if(k == 2) strings[i][k] = parsePercentage(agent.getAsString(new OID(globalOID + ".5." + (i+1))), agent.getAsString(new OID(globalOID + ".6." + (i+1))));;
        }
    }

    return strings;
}
```

Figure 3: Métodos para receção e aplicação dos dados recebidos através do *stdin*.

Por fim, e após serem guardados todos os valores, estes serão passados para a criação do HTML, sendo este, finalmente, passado para um `Printwriter`, que estará encarregue de o escrever em ficheiro.

```
public static String generateHTML(Object[][] array){
StringBuilder document = new StringBuilder();
    Date date = new Date();
    document.append("<!DOCTYPE html>\n");
    document.append("<html>\n");
    document.append("\n");
    document.append("    <head>\n");
    document.append("       <title>SNMP Table</title>\n");
    document.append("\n");
    document.append("<style>\n");
    document.append("   #snmptable {\n");
    document.append("       font-family: " + "Trebuchet MS" + ", Arial, Helvetica, sans-serif;\n");
    document.append("       border-collapse: collapse;\n");
    document.append("       width: 100%;\n");
    document.append("   }\n");
    document.append("\n");
    document.append("   #snmptable td, #snmptable th {\n");
    document.append("       border: 1px solid #ddd;\n");
    document.append("       padding: 8px;\n");
    document.append("   }\n");
    document.append("\n");
    document.append("   #snmptable tr:nth-child(even){background-color: #f2f2f2;}\n");
    document.append("\n");
    document.append("   #snmptable tr:hover {background-color: #ddd;}\n");
    document.append("\n");
    document.append("   #snmptable th {\n");
    document.append("       padding-top: 12px;\n");
    document.append("       padding-bottom: 12px;\n");
    document.append("       text-align: left;\n");
    document.append("       background-color: #4CAF50;\n");
    document.append("       color: white;\n");
    document.append("   }\n");
    document.append("</style>\n");
    document.append("   </head>");
    document.append("\n");
    document.append("   <body>\n");
    document.append("\n");
    document.append("<c><h3>Tabela monitorizada a cada 10 segundos</h3>\n");
    document.append("   <table id=" + '"' + "snmptable" + '"' + ">\n");
    document.append("   <th>Partition</th>\n");
    document.append("   <th>Total Size in GB</th>\n");
    document.append("   <th>Pct free</th>\n");
    for(int i = 0; i < array.length; i++){
        document.append("<tr>\n");
        for(Object elem : array[i]){
            if(elem == null);
            else document.append("<td>").append(elem.toString()).append("</td>");
        }
        document.append("</tr>\n");
    }
    document.append("</table>\n");
document.append("<h4>Última atualização feita em: ").append(date.toString()).append("</c>\n");
document.append("</body>\n");
document.append("</html>\n");

    return document.toString();
}

public static void createHTML(Object[][] values) throws FileNotFoundException, UnsupportedEncodingException{
    try (PrintWriter writer = new PrintWriter("tabela.html", "UTF-8")) {
        writer.println(generateHTML(values));
    }
}
```

Figure 4: Métodos para geração do código `HTML`.

Um exemplo final da execução de todo este sistema pode ser apresentado pela seguinte imagem:

Tabela monitorizada a cada 10 segundos

Partition	Total Size in GB	Pct free
C:\ Label: Serial Number 2615ba55	418.48	18.06
D:\	0.0	0
E:\ Label:PEN Serial Number 2329597	1.99	80.05
Virtual Memory	0.46	28.75
Physical Memory	0.25	23.36

Última atualização feita em: Sun Feb 04 22:18:50 GMT 2018

Figure 5: Tabela final.

Para este exemplo terá sido usado um período de atualização de 10 segundos, visto que, mesmo apesar de não haver grande atualização na unidade principal (o disco C:/), podem existir outros dispositivos cuja informação percentual poderá ser importante (como a memória virtual ou a memória física) ao ponto de ser necessária a atualização semi-constante mas não em tempo real.