Instituto Federal Goiano - Campus Ceres
Bacharelado em Sistemas de Informação
Prof. Me. Ronneesley Moura Teles

Andrey Silva Ribeiro
Jônatas de Souza Rezende
Lucas Pereira de Azevedo

Obter retorno através do comando Runtime.getRuntime().exec("comando")

Outubro
2017

Sumário

Runtime.getRuntime().exec("comando")

1 Como funciona?

A classe **Runtime** possui um método para fazer uma chamada ao sistema operacional e executar algum programa, que é o método **getRuntime().exec("comando")**. A mesma deve ser utilizada em conjunto com a classe **Process**.

Aconselha-se que seja pouco utilizado, pois ele gera uma dependência em relação ao sistema operacional em questão, perdendo a portabilidade.

2 Código exemplo - Java.

```
1  import java.io.BufferedReader;
2  import java.io.File;
3  import java.io.FileReader;
4  import java.io.FileWriter;
5  import java.io.IOException;
6  import java.io.InputStreamReader;
7
8  public class Pegar_retorno {
9
10     public static void main(String[] args) throws Exception {
11
12         try{
13
14             String resultado;
15
16             String n_arquivo = "resultado.txt";
17
18             String comando = "CMD /C Rscript d:\\desvio_padrao.R";
19
20             Process exec = Runtime.getRuntime().exec(comando);
21
22             BufferedReader retorno = new BufferedReader(new
                   InputStreamReader(exec.getInputStream()));
23
24             while ((resultado = retorno.readLine()) != null){
25                 FileWriter arquivo;
26                 try {
27                     arquivo = new FileWriter(new File(n_arquivo));
28                     arquivo.write("Resultado da operacao: ");
29                     arquivo.write(resultado);
30                     arquivo.close();
31                     FileReader reader = new FileReader(n_arquivo);
32                     BufferedReader leitor = new BufferedReader(reader);
33                     while(leitor.ready()) {
34                             System.out.println(leitor.readLine());
35                     }
36                     reader.close();
37                     leitor.close();
38
39                 } catch (IOException e) {
40                     System.err.println("Erro: " + e.toString());
41                 } catch (Exception e) {
42                     System.err.println("Erro: " + e.toString());
43                 }
44             }
45
46             System.out.println("\nResultado gravado em resultado.txt\n");
47             retorno.close();
48             exec.destroy();
49         }
```

```
50
51          catch(IOException exc){
52              System.err.println("Erro: " + exc.toString());
53          }
54      }
55  }
```

3 Código exemplo - R

Abaixo segue o código contido no script do **software R** que será usado no comando para obtenção do retorno:

```
1  x <- c(65, 72, 70, 72, 60, 67, 69, 68)
2
3  sd(x)
```

Executando o script acima o resultado apresentado é o seguinte:

```
1  [1] 3.97986
```

4 Explicação do código.

1. Importações necessárias para a execução do código Java:

```
1  import java.io.BufferedReader;
2  import java.io.File;
3  import java.io.FileReader;
4  import java.io.FileWriter;
5  import java.io.IOException;
6  import java.io.InputStreamReader;
```

2. Declaração das variáveis.

```
1  String resultado;
2  String n_arquivo = "resultado.txt";
3  String comando = "CMD /C Rscript d:\\desvio_padrao.R";
```

3. O método **exec** retorna um **Process** onde é possível obter a saída do programa e enviar dados para entrada, dentre outros.

```
1  Process exec = Runtime.getRuntime().exec(comando);
```

4. A classe **BufferedReader** serve para leitura de uma **InputStreamReader**, que por sua vez lê bytes e decodifica-os em caracteres, esses dados serão armazenados na variável **retorno**.

```
1  BufferedReader retorno = new BufferedReader(new InputStreamReader(exec
       .getInputStream()));
```

5. Se o retorno do método **exec** for diferente de nulo, a variável **resultado** recebe o valor que foi retornado, e a classe **FileWriter** que escreve dados em arquivos, pega esse retorno e salva no arquivo.

```
1  while ((resultado = retorno.readLine()) != null){
2      FileWriter arquivo;
```

6. Especifica o que vai ser escrito no arquivo de destino, ainda dentro da condicional. Na linha 1 será criado ou usado o arquivo, na linha 2 o texto que será escrito no arquivo, na linha 3 o resultado obtido a ser salvo no arquivo e na linha 4 finaliza o arquivo.

```
1  arquivo = new FileWriter(new File(n_arquivo));
2  arquivo.write("Resultado da operacao: ");
3  arquivo.write(resultado);
4  arquivo.close();
```

7. A classe **FileReader** executa a leitura dos dados do arquivo, cria a variável **reader** e armazena os dados obtidos na variável criada.

```
1 FileReader reader = new FileReader(n_arquivo);
```

8. A classe **BufferedReader** instancia a variável **leitor** onde armazena os dados lidos anteriormente. Na linha 3 é impresso na tela os valores obtidos enquanto tiver dados armazenados, na linha 5 encerra a variável **reader** e na linha 6 encerra a variável **leitor**.

```
1 BufferedReader leitor = new BufferedReader(reader);
2 while(leitor.ready()) {
3     System.out.println(leitor.readLine());
4 }
5 reader.close();
6 leitor.close();
```

9. Após toda a execução do código, apresenta na tela informando o nome do arquivo em que foi gravado os dados. Na linha 2 finaliza a variável **retorno** e na linha 3 encerra o método **exec**.

```
1 System.out.println("\nResultado gravado em resultado.txt\n");
2 retorno.close();
3 exec.destroy();
```

5 Referências

1. ORACLE. **Docs Oracle.** Disponível em: <https://docs.oracle.com/javase/7/docs/api/allclasses-noframe.html>. Acesso em: 5 de outubro de 2017.