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

Adallberto Lucena Moura
Andrey Silva Ribeiro
Anny Karoliny Moraes Ribeiro
Brener Gomes de Jesus
Daniel Moreira Cardoso
Davi Ildeu de Faria
Eduardo de Oliveira Silva
Gusttavo Nunes Gomes
Jônatas de Souza Rezende

Interação R com JSON

Outubro
2017

Sumário

Interação R com JSON

1 Introdução

Para propor formas de interação do Java com o R, primeiro precisamos entender o que é JSON e XML. XML, do inglês, eXtensible Markup Language e JSON, JavaScript Object Notation. Ambos, são formatos serializadores de dados organizados de forma hierarquica, como textos, banco de dados, ou desenhos vetoriais.

Nessa pesquisa apresentaremos o JSON, como escolha para realizar essa interação.

2 Por que JSON?

Após pesquisarmos sobre ambas as tecnlogias, principalmente em fóruns como Stack Overflow, GUJ e sites como GeekHunter, podemos observar que atualmente, o JSON vem sendo bastante utilizado. E para responder à dúvida de qual é melhor em determinada situação, na maioria dos tópicos encontrados sobre esse assunto, é comentado que para um projeto que possui um banco de dados relacional e bem estruturado, o JSON é a melhor opção. Por ser leve e bem intuitivo. Assim como no site oficial do JSON `http://json.org/json-pt.html`: "Para seres humanos, é fácil de ler e escrever. Para máquinas, é fácil de interpretar e gerar. JSON é em formato texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados."

2.1 Comparação JSON e XML

Código em XML:

```xml
<?xml version="1.0" encoding="utf-8"?>
<blog>
  <nome>zeno rocha - blog</nome>
  <url>http://blog.zenorocha.com</url>
  <post>
    <titulo>do it! - as simple as you can</titulo>
    <data>13/04/2011</data>
  </post>
  <post>
    <titulo>hello world</titulo>
    <data>12/04/2011</data>
  </post>
</blog>
```

recursos/XML.xml

Código em JSON:

```json
{
  "nome": "zeno rocha - blog",
  "url": "http://zenorocha.com",
  "posts": [
    {
      "titulo": "do it! - as simple as you can",
```

```
      "data": "13/04/2011"
    },
    {
      "titulo": "hello world",
      "data": "12/04/2011"
    }
  ]
}
```

recursos/JSON.json

3 Como utilizar o JSON?

Para gravar arquivos JSON em java, podemos utilizar uma diversidade de formas, porém após pesquisas em fóruns já citados, o GSON foi o mais recomendado. GSON nada mais é que uma biblioteca do Google, open source, inclusive disponível no Github em: `https://github.com/google/gson`.

Mas por que utilizar o GSON e não o JSON nativo?

O principal objetivo do GSON é:

- Prover uma interface simples para ler e exportar no formato JSON;
- Permitir que objetos pré-existentes e que não possam ser alterados sejam convertidos para e partir de JSON;
- Suporte ao generics do Java;
- Representação customizada de objetos;
- Suporte a tipos complexos de objetos.

Assim, eliminando a dificuldade do JSON para poucas linhas de código utilizando o GSON.

Para se utilizar o GSON, é preciso importar a biblioteca GSON para o projeto, que está incluída no arquivo desta pesquisa, em bibliotecas. E com facilidade utilizar o mesmo:

```
//Importacao da biblioteca
import com.google.gson.Gson;

//Criacao do objeto GSON
Gson gson = new Gson();

//Conversao de GSON para JSON para um arquivo .json
writeFile = new FileWriter("saida.json");
writeFile.write(gson.toJson(ent_valor.valores));
```

recursos/GSON.java

4 Como fazer a interação do R com o JSON?

Para utilizarmos o JSON no R, primeiramente precisamos instalar o pacote rjson, uma única vez com o código no console do R:

```r
install.packages("rjson")
```

recursos/rjson.R

Após isso é preciso dar o caminho do arquivo JSON para o R ler e armazenar como uma lista e calcular a mediana (exemplo que utilizamos).

```r
#informa a biblioteca
library("rjson")

#informa a pasta de origem do arquivo
setwd("C:\\Users\\qualq\\Documents\\GitHub\\estatisticaweb\\pesquisas\\
    interacao_R\\codigos")

# O arquivo e lido e os dados armazenados como uma lista.
result <- fromJSON(file = "saida.json")

# converter os dados para um quadro de dados R
result1 <- as.data.frame(result)

# converter texto em numero
aux <- apply(result1,1,as.numeric)

cat(median(aux))
```

recursos/calculo_mediana_usando_json.R

5 Como fizemos?

Primeiro criamos uma classe para Valores, que serão os valores que o usuário irá informar.

```java
package Aplicativo;

//classe criada para armazenar os valores informados no aplicativo
class Valor {

    String valor;

    public Valor(String valor) {
        this.valor = valor;
    }

}
```

recursos/Valor.java

A classe principal para ler esses valores e armazenar no JSON e chamar a classe referente ao sistema operacional utilizado

```java
package Aplicativo;

import com.google.gson.Gson;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

public class Aplicativo {
```

```
10
11      //arraylist criada para armazenar os valores recebidos para depois converter em JSON
12      ArrayList<Valor> valores = new ArrayList() ;
13
14      public static void main(String[] args) throws Exception{
15
16          //objeto criado para receber os valores que comporao a arraylist
17          Aplicativo ent_valor = new Aplicativo();
18          String valor;
19
20          Scanner entrada = new Scanner(System.in);
21          boolean continuar = true;
22
23          System.out.println("\nSeja bem vindo(a) ao sistema de calculo da mediana!!!");
24          System.out.println("\nInforme os valores desejado, para iniciar a execucao aperte ENTER no espaco vazio\n");
25          while(continuar){
26              valor = entrada.nextLine();
27              //condicional necessaria para verificar se o valor e vazio
28              if(valor.isEmpty())
29                  break;
30
31              ent_valor.valores.add(new Valor(valor));
32          }
33          //imprime os valores informados
34          /*for(int i = 0; i < ent_valor.valores.size(); i++){
35              System.out.println(ent_valor.valores.get(i).valor);
36          }*/
37
38          //criacao do objeto para converter para JSON
39          Gson teste = new Gson();
40          //exibe os dados no formato JSON
41          //System.out.println(teste.toJson(ent_valor.valores));
42
43          //cria o objeto para criacao do arquivo JSON
44          FileWriter writeFile = null;
45
46          try{
47              //informa o nome do arquivo a ser usado para gravar os dados no formato JSON
48              writeFile = new FileWriter("saida.json");
49              //informa que dados deverao ser gravados no arquivo
50              writeFile.write(teste.toJson(ent_valor.valores));
51              writeFile.close();
52          }
53          catch(IOException e){
54              e.printStackTrace();
55          }
56
57          //identifica qual o nome do sistema operacional executado na maquina
58          String resposta = System.getProperty("os.name");
59
60          //condicional para executar os comandos de acordo com o sistema operacional
61          if(resposta.contains("Windows"))
62              //executa os comandos contidos no metodo executar na classe
```

```
     Windows
63             Windows.executar();
64         else //como so temos windows e linux na faculdade, n fiz mais
     if, nao sendo windows, sera linux...
65             //executa os comandos contidos no metodo executar na classe
     Linux
66             Linux.executar();
67     }
68 }
```

recursos/Aplicativo.java

E na classe Windows, executamos o comando que executa o R no terminal com o código que apresentamos e em seguida pega o retorno, o resultado da função, grava em um arquivo TXT e exibe no terminal de execução o resultado o nome e local que o arquivo com o resultado foi gravado.

```
1  package Aplicativo;
2
3  import java.io.BufferedReader;
4  import java.io.File;
5  import java.io.FileReader;
6  import java.io.FileWriter;
7  import java.io.IOException;
8  import java.io.InputStreamReader;
9
10 public class Windows {
11
12     //criacao do metodo executar a ser usado na classe principal
13     public static void executar() throws Exception {
14
15         try{
16             // onde sera armazenado o retorno
17             String resultado;
18
19             // nome do arquivo que armazenara o resultado
20             String n_arquivo = "resultado.txt";
21
22             // o comando que sera utilizado
23             // deve obser o caminho do script em relacao ao computador
     que tiver sendo usado
24             // o Rscript devera estar habilitado no Path do windows,
     caso nao esteja basta informar o caminho completo do executavel
25             String comando = "CMD /C Rscript C:\\Users\\qualq\\
     Documents\\GitHub\\estatisticaweb\\codigos\\EstatisticaWeb\\web\\
     Script\\calculo_mediana_usando_json.R";
26
27             // executa o processo e armazena a referencia em 'exec'
28             Process exec = Runtime.getRuntime().exec(comando);
29
30             // pega o retorno do processo e armazena em buffer
31             BufferedReader retorno = new BufferedReader(new
     InputStreamReader(exec.getInputStream()));
32
33             // salva e imprime o retorno
34             while ((resultado = retorno.readLine()) != null){
35                 FileWriter arquivo;
36         try {
37                     // cria o arquivo para gravar os dados do retorno
38                     arquivo = new FileWriter(new File(n_arquivo));
39                     arquivo.write("\nResultado da operacao: ");
```

```
40                      arquivo.write(resultado);
41                      arquivo.close();
42                      // realiza a leitura do arquivo onde estao gravados
    os dados
43                      FileReader reader = new FileReader(n_arquivo);
44                      BufferedReader leitor = new BufferedReader(reader);
45                      // imprime o dado encontrado na tela
46                      while(leitor.ready()) {
47                          System.out.println(leitor.readLine());
48                      }
49                      reader.close();
50                      leitor.close();
51
52                  } catch (IOException e) {
53                      System.err.println("Erro: " + e.toString());
54      } catch (Exception e) {
55                      System.err.println("Erro: " + e.toString());
56      }
57              }
58              // pega o diretorio de trabalho atual
59              String dir = System.getProperty("user.dir");
60              // armazena o caminho completo do arquivo onde foi salvo os
    dados
61              dir = dir + "\\" + n_arquivo;
62              // imprime o caminho completo do arquivo
63              System.out.println("\nResultado gravado em " + dir + "\n");
64
65              retorno.close();
66              exec.destroy();
67          }
68
69          catch(IOException exc){
70              System.err.println("Erro: " + exc.toString());
71          }
72      }
73  }
```

recursos/Windows.java