Java e JSON

Professor: Vinícius Costa

JSON-P

- Java API for JSON Processing
- API padrão para processamento de objetos JSON
- API JSON-P permite trabalhar com JSON de duas formas:
  - Modelo de Objetos
    - Carrega os dados na memória em formato de árvore
    - Semelhante a API DOM do XML
  - Modelo de Streaming
    - Consome pouca memória pois faz o processamento a medida que vai lendo os dados
    - Semelhante a API SAX do XML

Criar JSON Utilizando Modelo de Objetos

1. Utilize um dos métodos estáticos de fabricação da classe JSON
   - Json.createObjectBuilder ( )
   - Json.createArrayBuilder ( )
2. Para criar um construtor, “builder” (objeto ou array), de um dos tipos:
   - JsonObjectBuilder
   - JsonArrayBuilder
3. Utilizando um objeto “builder” (objeto ou array) chame um dos metodos “add” para incluir os dados
4. Chame o método “build()” do objeto “builder” (objeto ou array) para criar o objeto de um dos tipos:
   - JsonObject
   - JsonArray

Criar JSON Utilizando Modelo de Objetos

- Exemplo

```
JsonObjectBuilder construtor= Json.createObjectBuilder();
construtor.add("nome", "José");
construtor.add("idade",30);
JsonObject objeto=construtor.build();
```

- Ou de forma abreviada

```
JsonObject objeto= Json.createObjectBuilder().
add("nome", "José").add("idade",30).build();
```

Métodos “get”

- Os métodos “get” do objeto do tipo “JsonObject” espera uma string por parâmetro, que representa a chave, e retorna o valor.
  - Exemplo

  variavel = objeto.get(“nome”);//um dos possíveis *get*
- Os métodos “get” do objeto do tipo “JsonArray” espera um inteiro por parâmetro, que representa o indice, e retorna o valor.
  - Exemplo

  JsonArray array;

  ...

  variavel = array.get(0); //um dos possíveis *get*

String para JSON com Modelo de Objetos

- Utilize o método estático “createReader( )” da classe Json para criar um “JsonReader”.
  - JsonReader leitor = Json.createReader(*parâmetro*);
- O método createReader pode receber como parâmetro qualquer objeto de classes que implementem “InputStream” ou “Reader”.
- Chame um dos métodos “read” (objeto ou array) do leitor para obter um “JsonObject” ou “JsonArray”
  - leitor.readObject()
  - leitor.readArray()

String para JSON com Modelo de Objetos

- Exemplo

```
String texto="{\"nome\":\"José\",\"idade\":30}";
InputStream fluxo=new ByteArrayInputStream(texto.getBytes());
JsonReader leitor=Json.createReader(fluxo);
JsonObject objeto= leitor.readObject();
```

- Ou de forma abreviada

```
JsonObject objeto=Json.createReader(new ByteArrayInputStream
("{\"nome\":\"José\",\"idade\":30}".getBytes())).readObject();
```

JsonReader e JsonWriter

- Ler texto e Transformar em Objeto JSON

```
String texto="{\"nome\":\"José\",\"idade\":30}";
StringReader fluxo=new StringReader(texto);
JsonReader leitor=Json.createReader(fluxo);
JsonObject objeto=leitor.readObject();
```

- Escrever JSON na Saída em Formato Texto

```
JsonWriter escritor=Json.createWriter(System.out);
escritor.writeObject(objeto);
```

Modelo de Streaming - Escrevendo JSON

- Criar uma fabrica de geradores de JSON utilizando o método estático Json.createGeneratorFactory
- Utiliza-se o método “createGenerator” da fabrica para criar um objeto do tipo “JsonGenerator”
- Escreve-se o objeto ou array raiz com os métodos “writeStartObject” ou “writeStartArray” respectivamente
- Utiliza-se os métodos “write” para continuar o fluxo do JSON. É possível ter objetos ou arrays internos chamando novamente um dos métodos “writeStart...”.
- Para finalizar um bloco de escrita utiliza-se “writeEnd”
- Para finalizar a escrita do JSON utiliza-se o método “close”

Modelo de Streaming - Escrevendo JSON

- Exemplo

```
JsonGeneratorFactory fabrica = Json.createGeneratorFactory(null);
JsonGenerator gerador = fabrica.createGenerator(out);
gerador.writeStartObject()
                .write("nome", "José")
                .write("idade",30)
                .writeEnd().close();
```

- Out pode ser do console, da servlet, etc

Modelo de Streaming - Lendo JSON

- Criar uma fabrica de analisador de JSON utilizando o método estático Json.createParserFactory
- Utiliza-se um dos métodos “createParser” da fabrica para criar um objeto do tipo “JsonParser”
- Com o parser verifica-se se existe informações a serem lidas com o método “hasNext”
- Para saber o tipo de infomação utiliza-se o método “next” do parser que retorna um objeto do tipo “Event”
- Com o objeto do tipo “Event” é possível saber qual é o tipo de informação e pega a informação com um dos métodos “get” do parser

Modelo de Streaming - Lendo JSON

```
String texto="{\"nome\":\"José\",\"idade\":30}";
StringReader fluxo=new StringReader(texto);
JsonParserFactory fabrica=Json.createParserFactory(null);
JsonParser analisador=fabrica.createParser(fluxo);
while(analisador.hasNext()){
    Event evento=analisador.next();
    if(Event.KEY_NAME==evento){
        System.out.print(analisador.getString());}
    if(Event.VALUE_STRING==evento){
        System.out.println(" :\""+analisador.getString()+"\"");}
    if(Event.VALUE_NUMBER==evento){
        System.out.println(" :"+analisador.getInt());}
}
```

Bibliografia

- http://www.matera.com/blog/post/java-ee-7-jsr353-trabalhando-com-objetos-json-em-aplicacoes-java
- https://www.infoq.com/br/news/2013/06/json-p-api-padrao-java
- https://docs.oracle.com/middleware/1213/wls/WLPRG/java-api-for-json-proc.htm#WLPRG1055