Manipulando arquivos Json com Java

Plain Old Java Objects - POJOs

Os POJOs - ou *Plain Old Java Objects* - são objetos Java com desenho simplificado. Em outras palavras, trata-se de um objeto ordinário, livre de quaisquer restrições especiais que não aquelas orientadas tão e somente pela especificação da linguagem Java (basicamente, as regras para a construção de classes). Assim, POJOs normalmente **NÃO** possuem:

1. Herança (não estendem outras classes);
2. Não implementam interfaces;
3. Não fazem uso de anotações (annotations)

Basicamente, um POJO representa uma entidade do domínio com o qual se trabalha. Abaixo, exemplifica-se um POJO possível para a representação de uma pessoa:

```
package br.unicamp.ic.mc322.model;

public class Pessoa {

  private String nome;
  private int idade;

  public String getNome() {
     return this.nome;
  }

  public void setNome(String nome) {
     this.nome = nome;
  }

  public int getIdade() {
     return this.idade;
  }

  public void setIdade(int idade) {
     this.idade = idade;
  }
}
```

POJOs possuem amplo uso para transferir dados entre diferentes componentes de uma aplicação, além de ter aplicação em diversos *frameworks*.

JSON

O JSON (*JavaScript Object Notation* - Notação de Objetos JavaScript) é uma formatação leve de troca de dados. Consistindo num sistema chave-valor, com cada uma de suas propriedades sendo separadas por vírgula, é de fácil legibilidade para seres humanos, enquanto para máquinas exige pouco poder de processamento para sua geração e interpretação, além de consumir reduzidos recursos quando trafegado pela rede. Abaixo, está representado uma instância da classe Pessoa, apresentada anteriormente, em notação JSON:

```
{
    nome : "Godofredo Dagoberto",
    idade : 56
}
```

Mais informações sobre a notação JSON pode ser encontrada através da documentação oficial do formato.

Manipulando JSON com uso da biblioteca Gson

A Google disponibiliza uma biblioteca de código aberto para a serialização e deserialização de objetos Json conhecida como Gson. Você pode fazer o download dos arquivos necessários para uso da biblioteca através deste link. Para tal, é importante especificar que é necessário fazer um mapeamento entre os objetos JSON e o domínio da aplicação: cada objeto JSON deverá possuir um POJO correspondente, com atributos que mapeiam as propriedades vistas no JSON.

1. Serialização

Supondo que deseja-se converter uma instância que representa uma pessoa, modelado de acordo com o POJO anteriormente apresentado para uma String, este objetivo pode ser facilmente atingido através do trecho de código apresentado abaixo:

```
Pessoa pessoa = new Pessoa();
pessoa.setNome("Godofredo Dagoberto");
pessoa.setIdade(56);

final Gson gson = new Gson();
String filme = gson.toJson(pessoa);
```

Após executado o procedimento, será gerada a String que representa a instância da classe pessoa, em formato Json, conforme mostrado abaixo:

```
{
```

```
    nome : "Godofredo Dagoberto",
    idade : 56
}
```

Essa String poderá ser salva em um arquivo de texto usando-se a API para manipulação de arquivos fornecida pela linguagem Java:

```
String arquivo = "{caminho para o arquivo onde se deseja salvar a informação}"
URL path = getClass().getClassLoader().getResource(arquivo);

try (FileWriter writer = new FileWriter(path.getPath(), false)) {
   writer.write(baseDados);
} catch (IOException e) {
   LOGGER.error("IOException: %s%n", e);
}
```

2. Deserialização

A deserialização representa o caminho inverso: a partir de uma String em formato Json, é possível fazer sua conversão para um POJO que representa aquela entidade. O primeiro ponto, normalmente, é fazer a leitura a partir de um arquivo de texto com uso da API Java. Uma vez que o arquivos tenha sua leitura concluída pela aplicação, a string resultante poderá então ser usada para para a obtenção de uma instância de uma classe que representa o objeto contido naquele arquivo Json:

```
String arquivo = "{caminho do arquivo json que contém o objeto}";
InputStream is = getClass().getClassLoader().getResourceAsStream(arquivo);
BufferedReader br = new BufferedReader(new
InputStreamReader(Objects.requireNonNull(is)));
{{Classe que representa o objeto}} fromJson = gson.fromJson(br, {{Classe que
representa o objeto}}.class);
```

{{Classe que representa o objeto}} é um POJO criado especificamente para o objeto Json com o qual se trabalha, contendo atributos com tipo e nome idênticos aos encontrados na String Json.