Apache POI: Manipulando Documentos em Java

Muitas empresas usam arquivos Excel para controlar vários processos. O framework mais conhecido para essa função na plataforma Java é o Apache POI, que é um framework open source que manipula diversos tipos de arquivo do Microsoft Office.

O **Apache POI** é um framework para a plataforma Java que possibilita a leitura e a escrita de dados em um documento do Microsoft Office. Com ele, é possível ler e escrever dados em arquivos Excel, Word, PowerPoint e arquivos de e-mail do Outlook. Existem ainda projetos para adicionar suporte para arquivos do Visio. O POI é bastante utilizado em frameworks conhecidos, como o PrimeFaces e o RichFaces, que tem funcionalidades como exportar dados de um DataTable diretamente para um arquivo Excel.

versão 3.10.1 que foi lançada em Julho de 2014. Já está disponível para download a versão 3.11.0 beta.

um arquivo de notas de alunos de uma turma, com os campos Nome, RA, Nota1, Nota2, Média e Aprovado. A partir desses campos serão calculados diversos dados para essa turma, como a média de notas da turma inteira, o nome do aluno com a maior e menor nota, o número total de alunos e o número de alunos reprovados e aprovados na disciplina. O Excel pode calcular esses dados, mas uma possível utilização dessa aplicação seria jogar todos os dados da planilha diretamente em um banco de dados, ou mostrar os dados da planilha em uma página web. Por isso, a manipulação de arquivos Excel em aplicações Java é bastante utilizada.

Configurando o Projeto

Para a configuração do projeto, será utilizado o Maven Framework, que facilita a importação das dependências do framework POI. A **Listagem 1** mostra o código do arquivo pom.xml do projeto: basta incluir a dependência do jar do framework. Nesse artigo usaremos a versão 3.10.1 que é a versão estável mais recente, que foi lançada em agosto de 2014.

Listagem 1. Configurando o POI com apoio do Maven.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/

        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache
        <modelVersion>4.0.0</modelVersion>
        <groupId>com.devmedia</groupId>
        <artifactId>teste</artifactId>
        <version>0.0.1</version>

        <dependencies>
```

```
14                      <version>3.10.1</version>
15                  </dependency>
16
</project>
```

Abrindo o Arquivo

O primeiro passo agora é criar o código que abre o arquivo Excel. **A Figura 1** mostra um exemplo do arquivo que será aberto. A primeira coluna do arquivo tem o nome de todos os alunos da disciplina, a segunda, o RA dos alunos, a terceira as notas da primeira prova, a quarta as notas da segunda prova e a quinta as medias dos alunos, que foi calculada pela formula (Nota1 + Nota2) / 2.

	A	B	C	D	E
1	Joao	123	5	7	6
2	Jose	234	1	6	3,5
3	Maria	435	6	8	7
4	Eduardo	765	9	6	7,5
5	Bruna	164	7	7	7
6	Luiz	765	8	6	7
7	Sonia	79	10	8	9

Figura 1. Arquivo Excel utilizado no desenvolvimento da aplicação do exemplo.

Para facilitar a manipulação dos dados do Excel, para cada linha do arquivo será criado um objeto do tipo Aluno, por isso foi criada a classe Aluno, que agrupa em seus atributos todos os dados que estarão no arquivo Excel. A **Listagem 2** mostra o código dessa classe.

Listagem 2. Código da classe Aluno.

```
1  package com.devmedia.teste;
2
```

```
6       private String ra;
7       private double nota1;
8       private double nota2;
9       private double media;
10
11
12      public Aluno() { }
13
14      public Aluno(String nome, String ra, double nota1, double nota2,
15                   double media, boolean aprovado) {
16            super();
17            this.nome = nome;
18            this.ra = ra;
19            this.nota1 = nota1;
20            this.nota2 = nota2;
21            this.media = media;
22            this.aprovado = aprovado;
23      }
24
25
26
27      public String getNome() {
28            return nome;
29      }
30      public void setNome(String nome) {
31            this.nome = nome;
32      }
33      public String getRa() {
34            return ra;
35      }
36      public void setRa(String ra) {
37            this.ra = ra;
38      }
39      public double getNota1() {
40            return nota1;
41      }
42      public void setNota1(double nota1) {
43            this.nota1 = nota1;
44      }
45      public double getNota2() {
46            return nota2;
47      }
48      public void setNota2(double nota2) {
49            this.nota2 = nota2;
50      }
51      public double getMedia() {
52            return media;
53      }
54      public void setMedia(double media) {
```

```
58        public boolean isAprovado() {
59              return aprovado;
60        }
61
63              this.aprovado = aprovado;
64        }
65  }
```

O *framework Apache POI* trabalha tanto com arquivos xls, com planilhas de versões antigas do Excel, como com arquivos xslx das versões mais recentes do Excel. O código para abrir um arquivo tanto de uma versão quanto da outra é bem parecido. O que muda é qual classe utilizar. Para versões antigas do Excel é utilizada a classe HSSFWorkbook e HSSFSheet, enquanto que para arquivos das novas versões do Excel são utilizadas as classes XSSFWorkbook e XSSFSheet, mas todo o restante do código é igual. No desenvolvimento desse artigo utilizaremos a classe HSSFWorkbook.

A sigla HSSF é uma piada da equipe de desenvolvimento do POI, por eles considerarem a formatação do arquivo do Excel muito ruim, a sigla HSSF significa Horrible SpreadShet Format. A sigla XSSF significa XML SpreadShet Format.

A **Listagem 3** mostra o código para abrir o arquivo Excel. O código apesar de grande é bastante simples. O primeiro passo é abrir o arquivo, com a classe FileInputStream, passando o PATH completo do arquivo. Depois utilizando a classe HSSFWorkbook, o arquivo é validado se é ou não um arquivo Excel. A classe HSSFSheet abre uma planilha específica do arquivo. O POI pode abrir diversas planilhas que estejam dentro de um arquivo Excel, no exemplo, como existe apenas uma planilha, é aberta a planilha com índice 0.

Depois de aberto o arquivo, e com a planilha que será processado aberta, é necessário ler célula a célula do arquivo, para isso, é recuperado um iterator

iterator, agora para iterar sobre as colunas de cada linha. Para ler as linhas do arquivo, é utilizada a classe Row, e para a célula especifica é utilizada a classe

A classe Cell possui diversos métodos para a manipulação dos dados do arquivo, por exemplo, é possível recuperar os dados que estão na célula com o tipo Java correto, por isso existem métodos para recuperar String, Números, Booleans entre outros. Também existem métodos para verificar qual o tipo de dado que está célula. Também é possível verificar se o valor da célula é calculado por uma formula.

Listagem 3. Código que abre e lê os dados do arquivo excel.

```
1  package com.devmedia.teste;
2
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.FileNotFoundException;
6  import java.io.IOException;
7  import java.util.ArrayList;
8  import java.util.Iterator;
9  import java.util.List;
10
11 import org.apache.poi.hssf.usermodel.HSSFSheet;
12 import org.apache.poi.hssf.usermodel.HSSFWorkbook;
13 import org.apache.poi.ss.usermodel.Cell;
14 import org.apache.poi.ss.usermodel.Row;
15
16 public class AbreExcel {
17
18        private static final String fileName = "C:/teste/teste.xls";
19
20        public static void main(String[] args) throws IOException {
21
22              List<Aluno> listaAlunos = new ArrayList<Aluno>();
23
24              try {
25                     FileInputStream arquivo = new FileInputStream(new File(
26                                   AbreExcel.fileName));
27
```

```
32                  Iterator<Row> rowIterator = sheetAlunos.iterator();
33
34                  while (rowIterator.hasNext()) {
                            Row row = rowIterator.next();
37
38                          Aluno aluno = new Aluno();
39                          listaAlunos.add(aluno);
40                          while (cellIterator.hasNext()) {
41                                  Cell cell = cellIterator.next();
42                                  switch (cell.getColumnIndex()) {
43                                  case 0:
44                                        aluno.setNome(cell.getStringCellValue()
45                                        break;
46                                  case 1:
47                                        aluno.setRa(String.valueOf(cell.getNume
48                                        break;
49                                  case 2:
50                                        aluno.setNota1(cell.getNumericCellValue
51                                        break;
52                                  case 3:
53                                        aluno.setNota2(cell.getNumericCellValue
54                                        break;
55                                  case 4:
56                                         aluno.setMedia(cell.getNumericCellValu
57                                        break;
58                                  case 5:
59                                        aluno.setAprovado(cell.getBooleanCellVa
60                                        break;
61                                  }
62                          }
63                  }
64                  arquivo.close();
65
66          } catch (FileNotFoundException e) {
67                  e.printStackTrace();
68                  System.out.println("Arquivo Excel não encontrado!");
69          }
70
71          if (listaAlunos.size() == 0) {
72                  System.out.println("Nenhum aluno encontrado!");
73          } else {
74                  double soma = 0;
75                  double maior = 0;
76                  double menor = listaAlunos.get(0).getMedia();
77                  int aprovados = 0;
78                  int reprovados = 0;
79                  for (Aluno aluno : listaAlunos) {
80                          System.out.println("Aluno: " + aluno.getNome() + " N
```

```
84                                  maior = aluno.getMedia();
85                          }
86                          if (aluno.getMedia() < menor) {
                                    menor = aluno.getMedia();
89                          if (aluno.getMedia() >= 6) {
90                                  aprovados++;
91                          }
92                          if (aluno.getMedia() < 6) {
93                                  reprovados++;
94                          }
95                  }
96                  double media = soma / listaAlunos.size();
97                  System.out.println("A media de notas e: " + media);
98                  System.out.println("A maior nota e: " + maior);
99                  System.out.println("A menor nota e: " + menor);
100                 System.out.println("O numero de alunos aprovados e: " + apr
101                 System.out
102                                 .println("O numero de alunos reprovados e: "
103                 System.out.println("Número total de alunos: " + listaAlunos
104             }
105
106     }
107 }
```

A **Figura 2** mostra o console do Eclipse com a saída para a execução do código, no console são exibidos o nome de todos os alunos que estão no arquivo Excel e suas médias, também são exibidos os cálculos da turma, como a média geral dos alunos, o número de alunos reprovados e aprovados, e o número total de alunos.

```
Aluno: Joao Média: 6.0
Aluno: Jose Média: 3.5
Aluno: Maria Média: 7.0
Aluno: Eduardo Média: 7.5
Aluno: Bruna Média: 7.0
Aluno: Luiz Média: 7.0
Aluno: Sonia Média: 9.0
A media de notas e: 6.714285714285714
A maior nota e: 9.0
A menor nota e: 3.5
O numero de alunos aprovados e: 6
O numero de alunos reprovados e: 1
```

Figura 2. Saída do console do Eclipse com os cálculos efetuados.

Além de ler o arquivo Excel, para cada linha do arquivo, é criado um objeto do tipo Aluno, esse objeto é adicionado em um ArrayList de alunos, e os dados dessa lista são analisados. Alguns cálculos realizados foram o cálculo da média das notas de todos os alunos que estavam no arquivo Excel, a maior e menor nota que estavam no arquivo, e o número de alunos aprovados e reprovados.

O Apache POI também possibilita a criação e a escrita de dados em um arquivo Excel, o código para isso também é bastante simples, e também existe a mesma diferença para arquivos XLS e XSLX apontada anteriormente. A **Listagem 4** apresenta o código para criar um arquivo Excel, caso ele ainda não exista, e escrever alguns dados de alunos no arquivo. Inicialmente são criados os dados de alguns alunos, e depois esses alunos são salvos no arquivo.

O primeiro passo, é criar o arquivo, com a classe HSSFWorkbook, e depois criar a planilha com a classe HSSFSheet, é possível nomear essa planilha, no exemplo, a planilha é chamada de Alunos. Depois de criada a planilha, é feito um for que itera sobre a lista onde os alunos estão cadastrados. A cada aluno que existe na lista, é criada uma linha no arquivo excel, e para cada célula dessa linha são adicionados os atributos do aluno, na mesma ordem do arquivo utilizado no exemplo para a leitura dos dados. Por fim, é dado um nome para o arquivo, utilizando a classe FIleOutputStream, e os dados do criados para a planilha são salvos no arquivo.

Listagem 4. Código que cria e salva os dados em um arquivo Excel.

```
1  package com.devmedia.teste;
2
3  import java.io.File;
4  import java.io.FileNotFoundException;
5  import java.io.FileOutputStream;
```

```
9
10 import org.apache.poi.hssf.usermodel.HSSFSheet;
11 import org.apache.poi.hssf.usermodel.HSSFWorkbook;
12 import org.apache.poi.ss.usermodel.Cell;
13
14
15 public class CriaExcel {
16
17       private static final String fileName = "C:/teste/novo.xls";
18
19       public static void main(String[] args) throws IOException {
20
21             HSSFWorkbook workbook = new HSSFWorkbook();
22             HSSFSheet sheetAlunos = workbook.createSheet("Alunos");
23
24             List<Aluno> listaAlunos = new ArrayList<Aluno>();
25             listaAlunos.add(new Aluno("Eduardo", "9876525", 7, 8, 0, false));
26             listaAlunos.add(new Aluno("Luiz", "1234466", 5, 8, 0, false));
27             listaAlunos.add(new Aluno("Bruna", "6545657", 7, 6, 0, false));
28             listaAlunos.add(new Aluno("Carlos", "3456558", 10, 3, 0, false));
29             listaAlunos.add(new Aluno("Sonia", "6544546", 7, 8, 0, false));
30             listaAlunos.add(new Aluno("Brianda", "3234535", 6, 5, 0, false));
31             listaAlunos.add(new Aluno("Pedro", "4234524", 7, 5, 0, false));
32             listaAlunos.add(new Aluno("Julio", "5434513", 7, 2, 0, false));
33             listaAlunos.add(new Aluno("Henrique", "6543452", 7, 8, 0, false));
34             listaAlunos.add(new Aluno("Fernando", "4345651", 5, 8, 0, false));
35             listaAlunos.add(new Aluno("Vitor", "4332341", 7, 9, 0, false));
36
37             int rownum = 0;
38             for (Aluno aluno : listaAlunos) {
39                 Row row = sheetAlunos.createRow(rownum++);
40                 int cellnum = 0;
41                 Cell cellNome = row.createCell(cellnum++);
42                 cellNome.setCellValue(aluno.getNome());
43                 Cell cellRa = row.createCell(cellnum++);
44                 cellRa.setCellValue(aluno.getRa());
45                 Cell cellNota1 = row.createCell(cellnum++);
46                 cellNota1.setCellValue(aluno.getNota1());
47                 Cell cellNota2 = row.createCell(cellnum++);
48                 cellNota2.setCellValue(aluno.getNota2());
49                 Cell cellMedia = row.createCell(cellnum++);
50                 cellMedia.setCellValue((aluno.getNota1() + aluno.getNota2()) /
51                 Cell cellAprovado =row.createCell(cellnum++);
52                 cellAprovado.setCellValue(cellMedia.getNumericCellValue() >= 6
53             }
54
55             try {
56                 FileOutputStream out =
57                         new FileOutputStream(new File(CriaExcel.fileName));
```

```
61
62              } catch (FileNotFoundException e) {
63                  e.printStackTrace();
64                  System.out.println("Arquivo não encontrado!");
65
66                  e.printStackTrace();
67                  System.out.println("Erro na edição do arquivo!");
68              }
69
70          }
71
72  }
```

A **Figura 3** mostra o arquivo Excel criado após a execução do código da **Listagem 4**. O arquivo foi criado com o nome definido na linha do FIleOutputStream, e o nome da planilha que foi definido na linha createSheet("Alunos").

	A	B	C	D	E	F
1	Eduardo	9876525	7	8	7,5	VERDADEIRO
2	Luiz	1234466	5	8	6,5	VERDADEIRO
3	Bruna	6545657	7	6	6,5	VERDADEIRO
4	Carlos	3456558	10	3	6,5	VERDADEIRO
5	Sonia	6544546	7	8	7,5	VERDADEIRO
6	Brianda	3234535	6	5	5,5	FALSO
7	Pedro	4234524	7	5	6	VERDADEIRO
8	Julio	5434513	7	2	4,5	FALSO
9	Henrique	6543452	7	8	7,5	VERDADEIRO
10	Fernando	4345651	5	8	6,5	VERDADEIRO
11	Vitor	4332341	7	9	8	VERDADEIRO

Figura 3. Arquivo Excel criado após a execução do código da **Listagem 4.**

Editando um arquivo Excel já existente

Outra possibilidade é abrir um arquivo já existente, e alterar os dados das células. Por exemplo, caso o professor queira adicionar um ponto na nota 1 de todos os alunos que estão no arquivo. É preciso alterar os dados da coluna Nota1, Média e também da coluna Aprovado. A **Listagem 5** mostra o código que abre o arquivo, e altera os valores das colunas citadas.

Na classe Cell, assim como existe os métodos get para recuperar os valores que estão nas células, existe também o método set, então para alterar os valores das

Listagem 5. Código que abre o arquivo Excel, e altera os dados das colunas.

```
package com.devmedia.teste;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;

public class EditaExcel {

	private static final String fileName = "C:/teste/novo.xls";

	public static void main(String[] args) throws IOException  {

		try {
			FileInputStream file = new FileInputStream(new File(
					EditaExcel.fileName));

			HSSFWorkbook workbook = new HSSFWorkbook(file);
			HSSFSheet sheetAlunos = workbook.getSheetAt(0);

			for (int i = 0; i < sheetAlunos.getPhysicalNumberOfRows(); 

				Row row = sheetAlunos.getRow(i);
				Cell cellNota1 = row.getCell(2);
				if (cellNota1.getNumericCellValue() < 9) {
					cellNota1.setCellValue(cellNota1.getNumericCe
				} else {
					cellNota1.setCellValue(10);
				}

				Cell cellNota2 = row.getCell(3);
				Cell cellMedia = row.getCell(4);
				cellMedia
```

```
43                              cellAprovado.setCellValue(cellMedia.getNumericCellVa
44                      }
45                      file.close();
46
47
48                      workbook.write(outFile);
49                      outFile.close();
50                      System.out.println("Arquivo Excel editado com sucesso!");
51
52              } catch (FileNotFoundException e) {
53                      e.printStackTrace();
54                      System.out.println("Arquivo Excel não encontrado!");
55              } catch (IOException e) {
56                      e.printStackTrace();
57                      System.out.println("Erro na edição do arquivo!");
58              }
59
60      }
61
62  }
```

A **Figura 4** mostra o arquivo Excel que foi criado no exemplo anterior, mas com os dados alterados pela execução do código da **Listagem 5**. A nota1 foi incrementada em 1 ponto para todos os alunos, com isso os dados da coluna média e da coluna aprovado foram alterados.

	A	B	C	D	E	F
1	Eduardo	9876525	8	8	8	VERDADEIRO
2	Luiz	1234466	6	8	7	VERDADEIRO
3	Bruna	6545657	8	6	7	VERDADEIRO
4	Carlos	3456558	10	3	6,5	VERDADEIRO
5	Sonia	6544546	8	8	8	VERDADEIRO
6	Brianda	3234535	7	5	6	VERDADEIRO
7	Pedro	4234524	8	5	6,5	VERDADEIRO
8	Julio	5434513	8	2	5	FALSO
9	Henrique	6543452	8	8	8	VERDADEIRO
10	Fernando	4345651	6	8	7	VERDADEIRO
11	Vitor	4332341	8	9	8,5	VERDADEIRO
12						

Figura 4. Dados alterados na planilha.

O Apache POI permite a manipulação completa de uma planilha Excel, esse artigo mostrou os comandos mais utilizados, mas ainda existem outras

dados para cada célula. Abrir diversas planilhas no mesmo arquivo. Além que o POI permite ainda abrir arquivos do Word, do PowerPoint e arquivos de e-mail do

Esse artigo mostrou como utilizar o framework Apache POI para a manipulação de arquivos do Microsoft Excel. Isso ainda é bastante utilizado, porque ainda hoje, muitas empresas ainda utilizam esse tipo de arquivo para o controle de diversas atividades. A aplicação desenvolvida neste exemplo mostra as principais funcionalidades do framework como abrir, criar e editar arquivos Excel.

Para mais informações sobre o projeto POI, o endereço do site do projeto é http://poi.apache.org/.

Espero que esse artigo seja útil! Até a próxima

Tecnologias:

Apache Java Maven XML

Por **Eduardo**
Em 2014

Suporte ao aluno - Deixe a sua dúvida.