Encoding e Charsets

Data	@21/02/2022
Tipo	📒Aula
Nivel	
Tópico	Encoding e Charsets

Encoding e Charsets

Nesta aula você aprendeu sobre Unicode, Encodings e Charsets.

Conheceu o problema dos Encodings: onde diferentes codepages são usados para escrever e exibir informações em seu computador. A solução foi dada for um consórcio que criou uma tabela genérica chamada Unicode contendo todos os caracters do mundo em números denominados codePoints. A segunda parte da solução é aplicar diferentes Encodings para definir como os bytes são guardados nos arqeuivos. Os encoding são tabelas que transformam cada codepoint em seu caracter especifico, dependendo de determinado região. Também observou que os encodings utilizados dependen muito de cada sistema operacional

Encoding com java.io

09)Se o *encoding* do **contas.csv** não estiver correto, você pode modificá-lo, clicando com o botão da direita do mouse sobre o arquivo e acessando ***Properties***. Em ***Text file encoding***, modifique o *encoding*. Faça o mesmo com os arquivos **lorem.txt** e **lorem2.txt**.

10) Abra a classe `TesteLeitura2`. Na instanciação do `Scanner`, passe o *charset* **UTF-8** como segundo parâmetro para o construtor:

```
Scanner scanner = new Scanner(new File("contas.csv"), "UTF-8");COPIAR CÓDIGO
```

11) Abra a classe `TesteLeitura`. Na instanciação do `InputStreamReader`, passe o *charset* **UTF-8** como segundo parâmetro para o construtor:

```
Reader isr = new InputStreamReader(fis, "UTF-8");COPIAR CÓDIGO
```

12) Abra a classe `TesteEscritaPrintStreamPrintWriter`. Na instanciação do `PrintWriter`, passe o *charset* **UTF-8** como segundo parâmetro para o construtor:

```
PrintWriter pw = new PrintWriter("lorem2.txt", "UTF-8");COPIAR CÓDIGO
```

13) Você também pode modificar o *encoding* do projeto modificá-lo, clicando com o botão da direita do mouse sobre o mesmo e acessando ***Properties***. Em ***Text file encoding***, modifique o *encoding*.

O que aprendemos

Usando o Windows, você implementou um programa para verificar a implementação do Java para Unicodes e Encodings e conheceu várias classes e métodos. Aprendeu que a classe String possui um método chamado `codePointAt()` para revelar o codepoint de determinado caractere a partir de sua posição na string. Descobriu que a classe que representa um encoding ou *Character Set* é `Charset` e o método estático para retornar uma referência com o charset default é `defaultCharset()`. Aprendeu que a classe String também possui um método para transformar os caracteres em bytes, o `getBytes()`, que usado sem argumento de entrada utiliza o charset padrão. Existem também duas sobrecargas para esse método, onde você pode informar o charset que deseja utilizar para a transformação. Conheceu a classe `StandardCharsets`, do pacote `java.nio`, que possui constantes pra os principais charsets. Por fim simulou o problema de encodings, gerando uma nova string a partir de um construtor que tinha como argumentos os bytes transformados no charset e o charset desejado para transformação. A solução foi garantir que o mesmo charset fosse aplicado, tanto na entrada quanto na saída.

As classes `Scanner` e `InputStreamReader` possuem sobrecargas de construtores que recebem como argumento qual charset será utilizado para fazer a transformação dos bytes em strings. De modo análogo para escrita, a classe `PrintWriter` também permite informar qual charset será utilizado para transformar a string nos bytes específicos.