13

Exibindo a Capa do Livro na Listagem

Transcrição

Depois de escrito no `outputChannel`, precisamos limpar o `buffer` para que ele possa ler novamente quando voltar ao fluxo do `while` que criamos.

Como já lemos novamente os bytes do `buffer` e entramos novamente no `while`, temos mais um passo a fazer. Todo buffer possui um ponteiro que após temos escrito no buffer pode ter ficado no final do mesmo, e para garantir que escreveremos no `outputStream` tudo que está no `buffer`, precisamos chamar o método `buffer.flip` que colocará o ponteiro na posição zero novamente. Assim, a primeira linha dentro do `while` será `buffer.flip()`

Nosso `while` completo ficou bem simples, graças ao `Channels` de entrada e de saída.

```
while(inputChannel.read(buffer) != -1) {
    buffer.flip();
    outputChannel.write(buffer);
    buffer.clear();
}
```

Além do `try-with-resources`, o `FileInputStream` nos obriga a verificar uma `Exception`, então faremos o `catch()` dele fora do nosso `try-with-resources` e apenas jogaremos a exceção pra cima, porém encapsulada em uma `RuntimeException()`.

Nosso método `transfer()` ficou assim:

```
public static void transfer(Path source, OutputStream outputStream) {
    try {
        FileInputStream input = new FileInputStream(source.toFile());
        try( ReadableByteChannel inputChannel = Channels.newChannel(input);
                WritableByteChannel outputChannel = Channels.newChannel(outputStream)) {
            ByteBuffer buffer = ByteBuffer.allocateDirect(1024 * 10);

            while(inputChannel.read(buffer) != -1) {
                buffer.flip();
                outputChannel.write(buffer);
                buffer.clear();
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}
```

Aparentemente temos tudo pronto, mas para que possamos ver de fato nossa capa, vamos abrir novamente nossa `lista.xhtml` e no lugar de exibir apenas o caminho da nossa capa, vamos exibir de fato a imagem de capa do livro, mas redimensionando para um tamanho que não quebre nossa lista de Livros.

Adicione a tag `<img />` e no atributo `src` coloque o valor como sendo `"#{request.contextPath}/file/#{livro.capaPath}". Ajuste o atributo altura para` height="30%" `e o texto alternativo como sendo o título do livro` alt="#{livro.titulo}"`. Assim, o código de exibição da imagem de capa ficou:

```
<h:column>
    <f:facet name="header">Capa Path</f:facet>
    <img src="#{request.contextPath}/file/#{livro.capaPath}"
            height="30%" alt="#{livro.titulo}" />
</h:column>
```

Faça novamente um *Full Publish* e cadastre um livro com uma imagem real e depois verifique na listagem se sua imagem realmente aparece corretamente na tela.

Ganhamos muita flexibilidade no nosso sistema, cadastrando as capas do livro e lendo esse arquivo de capa depois. Além disso, ainda fizemos a exibição dos arquivos como parte da URL, ao invés de ficar passando parâmetros, sujando nossas URL's do sistema.

Agora vá para os exercícios e tire suas dúvidas no fórum. Bons estudos!