12

Transferindo o arquivo do servidor para o response

Quase tudo pronto, só precisamos do método `transfer` da nossa classe `FileSaver` funcionando. Vamos lá.

Abra a classe `FileSaver` e nela crie um método público e `static`, que não deve possuir retorno chamado `transfer`. Os parâmetros que vamos passar são os que a classe `FileServlet` já usa. Por isso, receba um `Path` que chamamos de `source` e um `OutputStream` que chamamos de `outputStream`.

Dentro do método, vamos chamar o `new FileInputStream(source.toFile())` e já o colocar em uma variável. Agora temos que manipular os arquivos, e isso não é tão simples, por isso,segue abaixo o código do `try-with-resources`.

```
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);
}
```

Assim, esse código é bem tenso, e conforme explicamos no vídeo, ele pega o arquivo físico e passa para o `outputStream` do `HttpServletResponse`.

Com isso, fechamos o método `transfer`.