JasperReports e o *IOStream*

Guilherme Monteiro da Silva Lanna[1]
Adriano Franzoni Otavian[2]

Em sistemas web às vezes é interessante exibir ao usuário um arquivo PDF temporário antes de submetê-lo ao servidor. Este trabalho apresenta uma forma de fácil implementação e alta performance para tal questão usando a biblioteca de código aberto *JasperReports*.

Um pouco de conhecimento em programação *Java* (ORACLE, 2012) é suficiente para criar documentos em PDF por meio da biblioteca *JasperReports*. Embora seja necessário um arquivo *xml* para definir o modelo do documento, com o *iReport*, software disponibilizado pela *JasperForge*, sua criação é facilitada por uma interface visual bastante intuitiva, e, depois de pronto, já compila no formato *.jasper* que será usado na programação *Java* (JASPERSOFT, 2012).

Tratando-se de um sistema web, uma outra funcionalidade que pode ser alcançada é a criação de documentos temporários para serem exibidos na tela do navegador. Utilizando o método proposto basta que o programa tenha um arquivo *.jasper* disponível e atribua os parâmetros desejados dentro do código, em seguida o documento é enviado pela interface *IOStream* do *Java* para o usuário. Essa abordagem previne efeitos indesejados causados por concorrência em um servidor web assíncrono, além de prover um ganho de performance, pois não é necessário criar um documento temporário para depois exibi-lo em tela por meio de um comando *html* ou *javascript* que redireciona a janela para a *URL* onde o documento se encontra.

O primeiro passo para criar o documento é fazer o modelo no *iReport* que irá gerar um arquivo *.jasper*. Com esse arquivo pronto precisamos imple-

[1] Unicamp, gui.mslanna@gmail.com
[2] Embrapa Informática Agropecuária, adriano.otavian@embrapa.br

mentar o código *Java*. No projeto Agritempo foi construído um módulo, conforme a Figura1, com três classes: *Report.java*, uma classe com dois métodos principais implementados *gerarReportPDF()* e *visualizarReportPDF()*, *ReportBoletim.java*, classe filha do *Report.java* que unicamente implementa o construtor da classe pai especificando os caminhos do documento *.jasper* e de suas imagens, o caminho do local onde deve ser salvo o *report* e inicializa parâmetros necessários, *JasperReportsServlet.java*, classe filha da classe *HttpServlet.java* e implementa um único método *doGetReportBoletim()* que repassa os devidos parâmetros da requisição para um novo objeto *ReportBoletim* e invoca o método *gerarReportPDF()* ou *visualizarReportPDF()*.

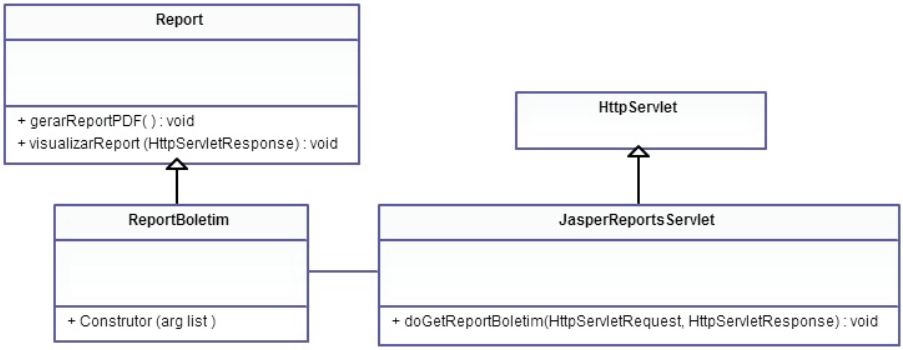

Figura 1. Diagrama de classes do módulo.

Da maneira que o módulo foi implementado, ele pode ser facilmente expandido para abranger outras especificações de relatórios e inclusive em outros formatos como *html* ou *xml* e ser usado em qualquer projeto.

Para exibir o relatório em tela, foi utilizado a interface *OutputStream* pelo método *response.getOutputStream()* e alterado o código do controlador json para utilizar o mesmo método no lugar do *response.getWriter()*, pois existe uma limitação que impede que o sistema utilize ambos os métodos ao mesmo tempo.

Para testar a performance, o sistema foi executado na máquina local por uso do *Tomcat* e utilizada a extensão *lori* (LORI, 2012) para o *firefox* que mede o tempo que uma determinada página leva para carregar. Gerando um arquivo *pdf* temporário de 356kB e redirecionando para uma página

intermediária com um contador regressivo que após um determinado intervalo de tempo carrega e exibe o arquivo, após vários testes com valores diferentes para o contador concluiu-se que era necessário uma margem de aproximadamente 6 segundos de espera para garantir que o documento seria exibido, enquanto, ao utilizar o *IOStream*, foi encontrado o valor médio de (1,60 ±0,09) segundos para exibir o mesmo arquivo de 356kB. A Figura 2 mostra o fluxo de execução dos dois modelos.

Figura 2. Fluxograma das etapas para exibir o relatório no navegador.

Concluiu-se que salvar arquivos temporários em disco para exibição imediata no navegador apresenta uma performance muito baixa para os padrões de internet atual, com a alternativa apresentada mostrou-se que é possível escalar performance sem grandes desafios de programação.

Referências

JASPERSOFT. 2012. Disponível em: <https://www.jaspersoft.com/>. Acesso em: 31 out. 2012.

LORI (Life-of-request-info). 2912. Disponível em: <https://addons.mozilla.org/en-us/firefox/addon/lori-life-of-request-info/>. Acesso em: 31 de out. 2012.

ORACLE. Disponível em: <http://docs.oracle.com/javaee/5/api/javax/servlet/ServletResponse.html>. Acesso em: 31 out. 2012.