UM MECANISMO PARA EXTERNALIZAÇÃO E FLEXIBILIZAÇÃO DE REGRAS DE NEGÓCIO

Marcius da Silva Fonseca[1], Marcos Vinícius B. de Souza[1], Giuliano Lopes Ferreira[1], Márcio Frick[1]
{sf.marcius, marcos, giuliano, mfrick}@cpd.ufsm.br
[1] Centro de Processamento de Dados da Universidade Federal de Santa Maria. Av. Roraima, 1000, Prédio 43, CEP 97105-900, Santa Maria, RS.

Palavras-chave: *Groovy*, Java, *Scripting*, Regras de Negócio.

1 INTRODUÇÃO

Durante o ciclo de desenvolvimento de sistemas podem ocorrer alterações em relação àquilo que fora inicialmente a idéia proposta, seja por questões de ordem técnica ou mudança nos processos de negócio. Estas alterações, na maioria das vezes, levam a modificações de código que acarretam a entrada em funcionamento de uma nova versão do sistema. Em um sistema que já se encontra em produção, tais mudanças levarão a ocorrência de um *redeploy* a cada nova alteração da aplicação. Muitos podem pensar, no entanto, que ao se efetuar uma mudança de regra de negócio, o código deverá ser alterado definitivamente e a aplicação recolocada no ar, pois se "*houve uma mudança em uma regra de negócio, então significa que esta estava errada anteriormente*". No entanto, em aplicações que funcionam no âmbito público, regras podem mudar com freqüência, seja por força de criação ou modificação de uma norma federal ou interna. Quando tais modificações acontecem, pode ser útil manter as versões das regras anteriores para posterior referência e até mesmo utilização em casos especiais. Desta forma, objetivando evitar a ocorrência de *redeploys* com freqüência, os analistas de sistemas da UFSM desenvolveram um mecanismo para que, na mudança de algumas regras de negócio, não seja necessário desempenhar este processo em uma aplicação inteira.

2 DISCUSSÃO E SOLUÇÃO

Durante a busca de soluções observou-se a existência de duas abordagens distintas ao problema: ***Business Rule Engines*** [1] (ou apenas *Rule Engines*) e ***Scripting Engines*** [2]. Ambas as abordagens foram analisadas e, ao final do estudo das possibilidades, optou-se pela *segunda*.

Para a implementação da solução, foi utilizada a linguagem Groovy [3] devido às seguintes características desta linguagem:

- Integração com java, havendo troca de informação em ambas as direções;
- Sintaxe de fácil compreensão e menos verbose;
- Possibilidade de uso tanto via compilação quanto interpretação em *runtime*;
- Facilidades na manipulação de coleções, expressões regulares, *Strings*, dentre outros;
- Apresenta suporte a estrutura "*closure*", tipagem estática e dinâmica;
- Possui facilidades de integração com *Spring Framework* 2.0;
- *Plugins* para Eclipse, IntelliJ Idea e NetBeans;
- Extensa documentação e material de estudo.

A implementação da solução envolveu a alteração do código-fonte original da *engine*/API *Groovy*. Esta foi realizada para que a API utilizasse uma espécie de *plugin* possibilitando a manipulação de códigos-fonte de classes contidas em um banco de dados ao invés de fontes contidos em sistema de arquivos. Como as funcionalidades originais da API *Groovy* não foram alteradas, a solução implementada, então denominada de Planilhas, mantendo todas as capacidades originais desta linguagem.

Esta arquitetura de Planilhas também foi integrada com a tecnologia EJB da seguinte maneira: Um EJB de serviços ao necessitar de uma dada planilha para efetuar a conclusão da sua tarefa solicita ao EJB de Carregamento de Planilhas o carregamento da mesma..O EJB de Carregamento de Planilhas,

então, acessa sua cachê de planilhas carregadas. Caso a planilha solicitada já tenha sido previamente utilizada, esta estará em cachê e será carregada sem a necessidade de compilação. Se a mesma ainda não foi utilizada ou sofreu alterações, esta será então compilada e (re)carregada a partir do banco de dados. Durante o processo de compilação (ou re-compilação) de uma planilha, é verificada a existência ou não da dependência para com outras planilhas. Caso haja, uma coleção de dependências é gerenciada e será compilada juntamente com a planilha solicitada.

Um exemplo de aplicação implementada e em uso é o emprego destas planilhas durante o processo de matrícula dos alunos admitidos no processo seletivo.Assim que um aluno é aprovado e o período de confirmação de vagas é finalizado, funcionários administrativos da UFSM efetuam a verificação da documentação dos alunos que concluíram a sua confirmação de vaga e, caso esteja tudo de acordo com as normas do concurso da instituição, efetuam a matrícula do aluno no sistema acadêmico da universidade.Este processo de matrícula envolve a geração de um número de matrícula único para cada aluno. Cada tipo de concurso (Vestibular, PEIES, EAD, etc) possui uma forma específica de geração deste número e, freqüentemente esta forma de geração do número de matrículas é modificada.Por este motivo, este processo de geração de matrículas é processado por uma planilha específica para este fim. Ainda, as versões anteriores deste processo de geração do número de matrícula são mantidas no banco de dados como versão inativa, permitindo assim a consulta e até mesmo seu uso em casos especiais.

Além do desenvolvimento do mecanismo para acessar e processar as planilhas (scripts *Groovy*) também foi desenvolvido uma ferramenta para facilitar a criação das mesmas. A aplicação consiste em um portal Web para gerenciar tais planilhas. Este portal conta com uma tela de edição, contendo os campos necessários ao cadastro das planilhas, assim como uma tela de pesquisa e listagem destas onde é possível procurar e abrir para edição uma dada planilha e também, obter uma listagem das planilhas que atualmente estão ativas no sistema. O portal conta também com uma tela de ajuda, contendo um link para um manual no formato PDF, assim como um link para uma futura documentação no formato Wiki.

3 CONCLUSÃO

O requisito de externalizar as regras de negócio tem por objetivo facilitar a personalização da solução para os sistemas com necessidades distintas. Além disso, estas regras de negócio devem ser apresentadas em uma linguagem que as torne fácil de escrever, entender e manter pelos experts de processos de negócio, pessoas estas que muitas vezes não possuem o domínio técnico de programação de computadores, muito embora possuam um pensamento lógico avançado.

O mecanismo implementado mostrou-se efetivo para a resolução do problema em questão. A performance da execução das planilhas foi satisfatória e com o advento do portal de gerenciamento criado, a manutenção destas regras externas tornou-se mais acessível aos administradores dos diversos sistemas que as utilizam.

REFERÊNCIAS

[1] Sun Developer Network. "*Getting Started With the Java Rule Engine API*". Disponível em: <http://java.sun.com/developer/technicalArticles/J2SE/JavaRule.html>. Acesso em 14 de abril de 2010.
[2] Sun Developer Network. "*Scripting for the Java Platform*". Disponível em: <http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting>. Acesso em 14 de abril de 2010.
[3] CodeHaus Foundation. "*Groovy – Technical Documentation*". Disponível em: <http://groovy.codehaus.org/Documentation>. Acesso em 14 de abril de 2010.