Desenvolvimento de Uma API de Acesso às Informações dos Serviços Publicados no JBoss ESB

Nailson dos Santos Cunha[1], Jônatas Clementino Melo[1], Pablo Sebastian Veinberg[1], Thiago José Marques Moura[1]

[1]Unidade Acadêmica de Informática
Instituto de Tecnologia da Paraíba – (IFPB) – João Pessoa, PB – Brazil

{nailsoncunha, jonatas.cmelo, pveinberg, tmoura}@gmail.com

Abstract. *This work was part of a research project developed with the Project SIM, City Hall of João Pessoa and IFPB. It had as goal to develop an API to access the information services deployed on JBoss ESB and can be used for any project that has this ESB as an integration tool. The development of this API made use of free and open source technologies.*

Resumo. *Este trabalho foi parte integrante de um projeto de pesquisa desenvolvido junto ao Projeto SIM, Prefeitura Municipal de João Pessoa e IFPB. Teve como objetivo desenvolver uma API de acesso as informações dos serviços implantados no JBoss ESB, podendo ser utilizada por qualquer projeto que tenha esse ESB como ferramenta de integração. O desenvolvimento dessa API fez uso de tecnologias livres e de código aberto.*

1. Introdução

A criação da API teve como motivação um problema real que surgiu no desenvolvimento de um Sistema de Informação Municipal para a Prefeitura Municipal de João Pessoa (PMJP), esse projeto foi intitulado Projeto SIM. O Projeto SIM foi criado com o objetivo de desenvolver sistemas para todos os setores da prefeitura. Era necessário integrar alguns sistemas legados em funcionamento, por enquanto que não eram substituídos, com os novos sistemas em desenvolvimento através de um *Enterprise Service Bus* (ESB). O ESB escolhido foi o JBoss ESB 4.11 de código aberto e que atendia completamente os requisitos do projeto além de executar no servidor de aplicação JBoss AS (*Application Server* 5.1.0), o mesmo utilizado pelo SIM.

Com a utilização do ESB, os arquitetos do SIM levantaram a problemática do desenvolvimento colaborativo de serviços. Diversos times iriam criar e publicar serviços, sendo necessário que eles consultassem esses serviços já em execução no JBoss ESB, para que pudessem verificar além de nome e descrição, informações de como consumí-los e quais as suas funcionalidades.

Outro objetivo era disponibilizar informações de forma automática sobre todos os serviços do JBoss ESB que pudessem ser acessados por softwares de terceiros (contratos externos da prefeitura), em uma espécie de catálogo de serviços. Dada essa problemática, o artigo em questão traz a solução encontrada: uma API de código aberto responsável por extrair informações de todos os serviços publicados no JBoss ESB.

2. Tecnologias envolvidas

Antes de prosseguirmos, devemos entender alguns conceitos e ferramentas que foram utilizados para o desenvolvimento desse trabalho. O mais importante conceito é o ESB, conector que liga múltiplos sistemas ou subsistemas. Ele se localiza em uma camada arquitetural e trabalha sobre um sistema de mensagens que permite comunicação síncrona ou assíncrona entre os serviços implantados nele [DIMAGGIO, et al. 2012]. Ou seja, um ESB é uma plataforma de integração, baseada em padrões, que combina mensagens, *web services*, transformação de dados e roteamento inteligente entre organizações e seus parceiros de negócios, separados por fronteiras e limites físicos, com integração transacional [Chappell 2004].

O JBoss ESB, como dito anteriormente, é um ESB de código livre desenvolvido pela JBoss. A implantação de novos serviços se dá, principalmente, através da configuração de arquivos XML. O arquivo "jboss-esb.xml" é o principal arquivo de configuração do JBoss ESB e onde todos os serviços são configurados com informações, por exemplo de *endpoint* (ponto de acesso ao serviço), nome do serviço, descrição.

É preciso entender que no JBoss ESB pode-se fazer *deploy* de vários projetos e cada um deles possui um arquivo XML próprio contendo múltiplos serviços. Os projetos podem ser implantados de duas formas: um arquivo de extensão esb, similar ao formato *Java Archive* (JAR) ou uma pasta com um nome qualquer adicionando ao seu final o ponto esb (.esb). Assim, essas situações diferentes tinham de ser levadas em conta e a API deveria fornecer um forma de acesso padrão, independente da maneira como o projeto se encontra dentro do ESB.

A forma de acessar o JBoss ESB e conseguir extrair os serviços disponíveis foi através do servidor JMX que é implementado e iniciado no JBoss AS. O JMX (*Java Management Extensions*) é uma tecnologia Java que fornece ferramentas para gerenciamento e monitoramento de aplicações, objetos de sistema, dispositivos e redes orientadas a serviço. É uma API que usa o conceito de agentes, permitindo monitorar elementos da Máquina Virtual Java e dos aplicativos que estão sendo executados. Para gerenciar recursos usando a tecnologia JMX são usados objetos Java conhecidos como *Managed Beans*, ou MBeans, que são os responsáveis por implementar o acesso à instrumentação de recursos [Sullins e Whipple 2003].

3. Solução Proposta: API de acesso aos serviços do JBoss ESB

Antes da implementação da API, alguns requisitos tiveram de ser seguidos:

- Obter informações de nome, descrição e tipo do serviço. No caso de *web services* também é necessário encontrar a URL do arquivo WSDL;
- Fornecer a opção de definição da url de conexão com o servidor JMX do JBoss AS, uma vez que o endereço pode ser modificado nas configurações do servidor;
- Não permitir alteração direta no arquivo jboss-esb.xml. O objetivo da API é apenas leitura das informações;
- Não acessar os serviços do núcleo do JBoss ESB. O JBoss ESB possui implantados alguns serviços necessários ao seu funcionamento, esses tem características diferentes dos que são criados pelos desenvolvedores;

3.1. Mapeamento do jboss-esb.xml para Classes Java

Foi utilizada a API de anotações do XStream para representar os relacionamentos que existem entre as tags XML do aquivo jboss-esb.xml. As principais tags utilizadas na configuração e publicação dos serviços no jboss-esb.xml foram mapeadas para classes java. O XStream é uma biblioteca Java, de código aberto que tem como objetivo facilitar a serialização de objetos Java para XML ou vice-versa através de reflexão. Funciona com anotações Java para casos mais complexos de transformação, que envolvam grandes arquivos XML com vários atributos, ou até objetos Java com muitos relacionamentos [XStream 2013].

Vale salientar que, por questões de espaço, não será aqui exibido o diagrama de classes que representa as tags do arquivo jboss-esb.xml. O diagrama é extenso, já que contempla diversas tags utilizadas no arquivo XML. Essas classes estão anotadas utilizando a API de XStream, para através de reflexão, em tempo de execução, criar os objetos Java equivalentes as informações encontradas no jboss-esb.xml.

3.2. Implementação da API

Após o mapeamento de todas as tags XML do arquivo jboss-esb.xml em classes Java, foram implementadas outras três classes responsáveis pela acesso ao servidor JMX, leitura do arquivo jboss-esb.xml e transformação das tags em objetos Java, utilizando as classes já mapeadas. O diagrama da figura 1 representa as principais classes de processamento da API.

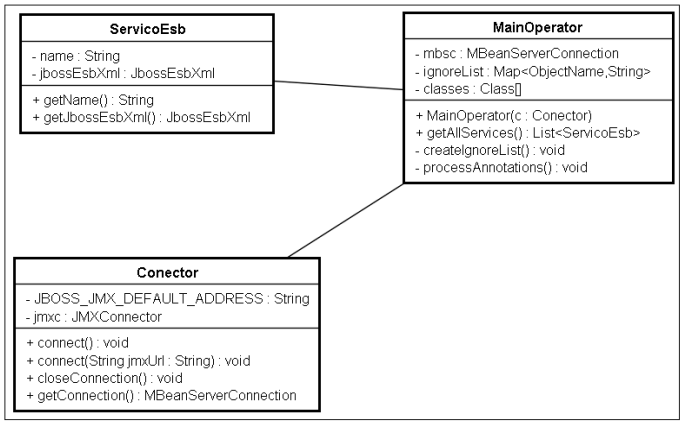

Figure 1. Principais classes desenvolvidas

A classe MainOperator é a principal classe de procesamento, responsável por retornar uma lista que contém todos os serviços disponibilizados no JBoss ESB, excetuando-se aqueles que são padrões do próprio JBoss ESB, como informado em um dos requisitos para elaboração da API.

A classe ServicoEsb tem o objetivo de representar o projeto ESB, com seus serviços, que está implantado no servidor. O método getJbossEsbXml() retorna um objeto JBossEsbXml que representa o mapeamento do arquivo jboss-esb.xml conforme explicado no tópico 3.1.

A classe Conector é responsável por se conectar ao servidor JMX. A classe possui o endereço padrão do servidor JMX, podendo ser alterado.

A figura 2 mostra o fluxo de execução de todo o processamento para acesso aos serviços do JBoss ESB utilizando a API desenvolvida.

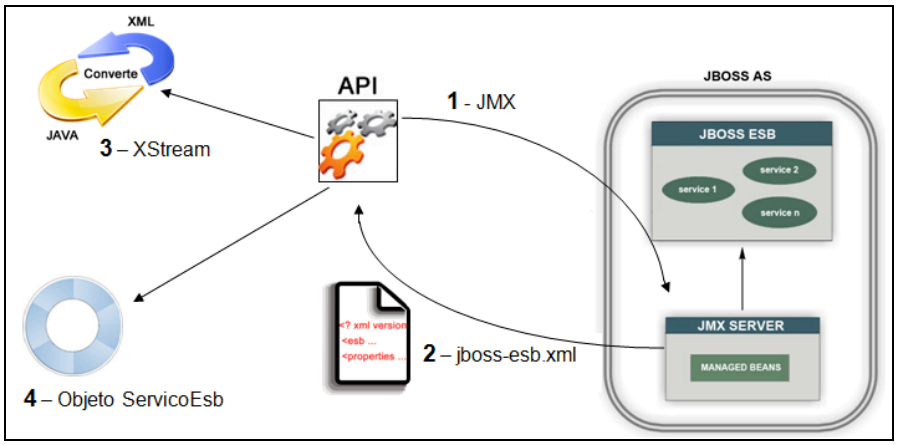

Figure 2. Fluxo de processamento

1) Na criação do objeto MainOperator, um objeto Conector é passado como parâmetro e esse é responsável por se conectar ao servidor JMX através do endereço de acesso;

Dentro do método getAllServices() os outros passos são executados:

2) Conexão realizada com o servidor JMX, carrega todos dos arquivos jboss-esb.xml existentes;
3) O método processAnnotations() é chamado e lê cada arquivo XML, transforma-os em objetos Java, de acordo com o mapeamento explicado no tópico 3.1;
4) Retorna uma lista de objetos ServicoEsb, excetuando-se os que são padrões do próprio JBoss ESB.

4. Conclusão

A principal dificuldade encontrada foi conseguir se conectar ao servidor JMX implementado no JBoss AS, já que não sabíamos qual o endereço de acesso, além de entender a API para extrair as informações que buscávamos. Superadas as dificuldades, esse trabalho foi importante para a continuação do Projeto SIM junto a PMJP. Será construída uma aplicação web, que utilize a API, para que os desenvolvedores possam verificar quais serviços estão disponíveis no JBoss ESB, consultando informações desses serviços e assim, evitando de implementarem novos serviços com as mesmas finalidades dos já existentes.

Referências

Chappell, David A. (2004), Enterprise Service Bus: Theory In Practice, O'Reilly Media, 1ª. edição.

Dimaggio, L., Conner, K., Kumar, M. and Cunningham, T. (2012), JBoss ESB: Beginner's Guide, Packt Publishing, 1ª edição.

Sullins, B. G.; Whipple, M. B. (2003) JMX in Action, Manning Publications, 1ª edição.

XStream (2013). "About XStream", http://xstream.codehaus.org/index.html, Março.