Reflexão 03 - APIs de BDs

Paulo Kiyoshi Oyama Filho

“A reflexão desta semana é sobre o uso de frameworks para BD relacionais. O objetivo é que você discuta, preferencialmente lendo opiniões de programadores em diferentes fontes, sobre as razões de JPA (em particular Hibernate e outros) terem tido sucesso e potencialmente serem preferíveis (ou não) a JDBC.”

O Java DataBase Connectivity (JDBC) possui a função de ser uma camada de abstração para com uma parte importante no desenvolvimento que é o armazenamento dos dados por meio de um Banco de Dados, ele usa de um gerenciamento para conseguir migrar um código para diferentes tipo de BD’s, contudo o que o torna menos preferível em relação ao Java Persistence API (JPA) são pequenos fatores dos quais falaremos mais adiante.

Primeiramente, o uso direto de códigos SQL para o contato com o banco de dados, o framework JDBC necessita do uso de códigos SQL para funções básicas como criação de uma tabela e movimentação de operações báscias como o C.R.U.D, enquanto que o JPA possui uma camada de abstração acima, com a qual faz essa implementação de maneira discreta ao desenvolvedor, o código sabe que deve gerar uma tabela ou realizar um operação de inserção por meio de funções ou notações dados por esse framerwork, como por exemplo, um @Table no código em Java, além disso, algo que pode ser um fator decisivo é a questões da afinidade do programador Java com a linguagem SQL, se caso ele não souber torna-se preferível o JPA já que ele “esconde” o SQL em questão.

Outra questão, é o fato do código misturado, muitos programadores podem achar o uso de uma parte do arquivo com códigos em SQL e outros em Java um tanto incômodo, haja vista algumas experiências em PHP.

Ainda sim, temos a questão das transações, no JDBC a questão das querys, transações e os rollbacks devem ser tratados pelo próprio desenvolvedor Java, o que para uns é muito complicado ou não sabem implementar, já no JPA, a camada de abstração dita acima cuida da questão das transações por baixo do pano. Outra vez que essa camada de abstração acima do JDBC torna-se mais chamativa é a questão das Exceções, no código JPA, as exceções do banco de dados são tratadas pelo framework enquanto que no JDBC tem a necessidade de blocos try, catch para pegar essas Exceções e tratá-las de forma efetiva.

Por fim, pelas minhas pesquisas, vi algumas vantagens no JDBC e alguns pontos importantes, por exemplo, muitos lugares argumentam que o JPA é um framework que usa das chamadas do JDBC embaixos do pano, por esta causa disso que argumento que ele é uma camada de abstração acima do sistema de gerenciamento de banco de dados oferecido pelo JDBC, por isso o uso de JDBC traria um contato mais “direto” com o banco de dados eliminando essa camada de abstração que o JPA coloca, deixando ao programador mais visível do que está acontecendo. Além disso, o uso de JDBC torna-se mais interessante para programadores que são acostumados com banco de dados e códigos em SQL, haja vista que a implementação do próprio código em SQL atende, de uma maneira mais certeira, a necessidade do programador que a está criando enquanto que uma notação já implementada pode não ser o que ele quer ou, ás vezes, não contemplar todas as suas necessidades.