Instituto Federal Goiano - Campus Ceres
Bacharelado em Sistemas de Informação
Prof. Me. Ronneesley Moura Teles

Davi Ildeu de Faria
Igor Justino Rodrigues
Luciano de Carvalho Borba
Warley Rodrigues de Andrade
Wesley Morais Felix

JPA

Java Persistence API

Novembro
2017

Sumário

JPA

1 Introdução

Com o Java tornando-se cada vem mais populares em ambientes corporativos, observou-se que grande parte do tempo era gasto na codificação de consultas SQL ao banco de dados e no código JDBC que utiliza estas, além das diferenças SQL entre os SGBDs, e a mudança de paradigma, com isso surgiram ferramentas para auxiliar nesta tarefa, entre elas está a JPA.

2 O que é?

A Java Persistence API (JPA) é um framework que foi criada por um grupo de conhecedores de software EJB 3.0 para fazer parte do JSR 220, foi apoiada em POJOS (Plain Old Java Objects). A JPA não é apenas um framework para Object-Relational Mapping - ORM (método usado para reduzir a disparidade da Programação Orientada a Objetos utilizando banco de dados relacionais), ele também tem diversas funções essenciais em qualquer aplicação organizacional.

Atualmente, praticamente a maioria das grandes aplicações usam JPA para persistir objetos java. A versão 2.0 adicionou muitas características que não haviam na primeira versão, principalmente as mais pedidas pelos usuários, dentre estas está a capacidade adicional de mapeamento e crescimento para a (JPQL) Java Persistence Query Language.

A JPA cuida aproximadamente oitenta por cento dos problemas quando necessita integrar com bancos de dados diferentes, ele assegura a portabilidade da aplicação. A JPA define um modo para mapear Plain Old Java Objects(POJOs), este é utilizado para dizer que é um objeto Java comum, a Java Pesistence API define o esquema entre o objeto java e o BD usando ORM.

A última versão é a 2.1.

3 O que é necessário para funcionar?

Os drivers tem como base a tecnologia JDBC e são divididos em 4 categorias: Os drivers do tipo 1, podem ser utilizados sempre que não houver um driver específico para um determinado banco de dados:

1. **JDBC-ODBC + ODBC driver;**
   Java acessa o banco através de drivers ODBC. O driver deve ser carregado em cada cliente que realiza acesso ao banco.

2. **Driver Java com API-Nativa;**
   Neste caso as chamadas JDBC são convertidas diretamente em chamadas para a API dos banco de dados. Neste caso também é necessário que um código binário específico esteja presente no cliente.

3. **Driver Java Puro, JDBC-Net;**
   Este driver traduz chamadas JDBC em chamadas para um protocolo de Rede/DBMS independente que em seguida é traduzido para o DBMS por um

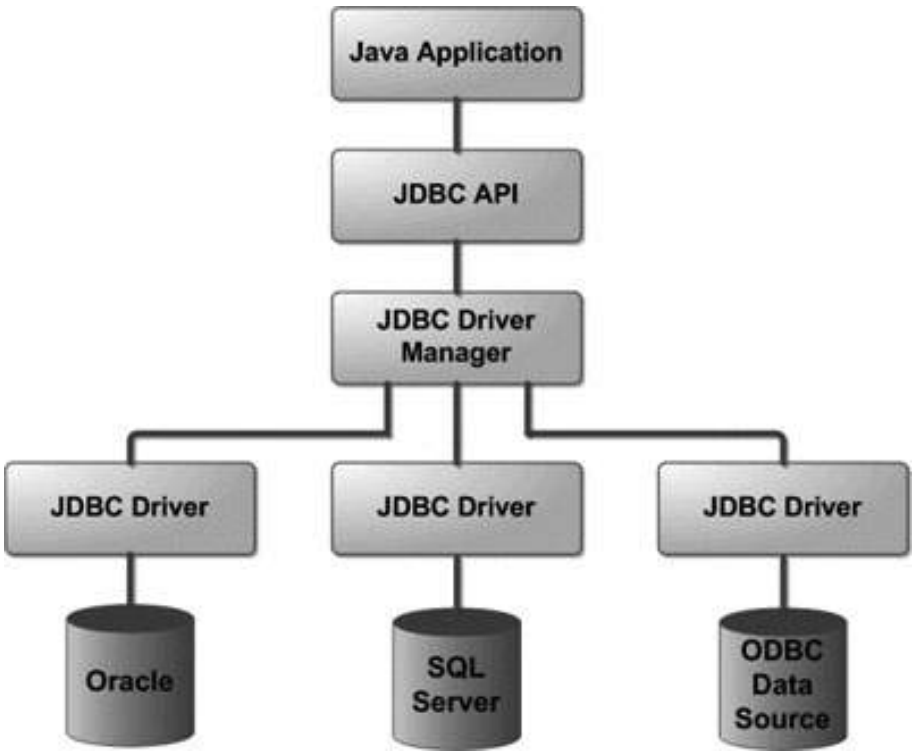

Figura 1: https://www.tutorialspoint.com/jdbc/images/jdbc-architecture.jpg

servidor. Este Middleware permite que cliente java puros se conectem com diferentes BD.

4. **Protocolo Nativo - Driver Java Puro;**
   Neste caso as chamadas JDBC são convertidas diretamente para o protocolo utilizado pelo DBMS, permitindo uma chamada direta do cliente para o servidor. A maioria destes drivers é proprietário.

4 Como utilizar?

Os seguintes passos são necessários se conectar uma aplicação Java com um banco de dados.
1. Criar a conexão ODBC com o BD (apenas no caso JDBC-ODBC);
2 Carregar o Driver;
3. Criar a Conexão com o BD;
4. Criar os comandos SQL;
5. Processar os comandos;
6. Finalizar a conexão com o banco de dados;
Os passos de 2 a 6, são executados diretamente no código java.

5 Diferenças do JDBC

- Java Procedure Query Language (JPQL) é uma linguagem de consulta ORM que trabalha em cima de classes e objetos, ja o SQL que funciona sobre tabelas.
- É uma liguagem bem semelhante a SQL.

6 Vantagens

- independência.
- Não há necessidade de ficar preso a detalhe específicos do SGBD.
- Reduz a necessidade de conhecer SQL.
- Otimização automática, uma consulta otimizada é um ponto essencial.

7 Desvantagens

- Grande quantidade de código para comandos sql em chaves compostas.
- A independência facilmente quebrada.

8 Conclusão

Visto que hoje em dia o tempo é essencial quando se está desenvolvendo um software, então a JPA é uma forma para poder ganhar tempo, além de poder dispensar um certo conhecimento com a linguagem SQL.

9 Referências

1. MEDEIROS, Higor. **Introdução à JPA - Java Persistence API.** Disponível em: <https://www.devmedia.com.br/introducao-a-jpa-java-persistence-api/28173>. Acesso em: 08 de novembro de 2017.
2. CAELUM. **Uma introdução prática ao JPA com Hibernate.** Disponível em: <https://www.caelum.com.br/apostila-java-web/uma-introducao-pratica-ao-jpa-com-hibernate/>. Acesso em: 08 de novembro de 2017.