Arquitetura de Software
ORM

Projeto de Software
Alphalinc Upgrade
Agosto/2014

TASC (the alpha supply chain)

1. Índice

2. Histórico de Revisões

Data	*Versão*	*Autor*	*Descrição*
31/07/14	1.00	Alexandre van den Mosselaar	Criação do documento

3 Introdução

3.1 Objetivo

Este documento tem como objetivo apresentar a arquitetura de software adotada para implementação do módulo de ORM do novo sistema Alphalinc (ALUP), conforme contemplado no projeto de migração do sistema para tecnologia Java com utilização de um banco de dados relacional (RDBMS). O módulo de ORM possibilita o acesso aos dados relacionais através uma interface orientada a objetos, compatível com os recursos da tecnologia sendo empregada. Além da arquitetura adotada, este documento apresenta os principais recursos da API que será utilizada pelos desenvolvedores, os métodos de acesso disponibilizados para compatibilização com o legado, a estrutura de modelagem e de mapeamento dos dados, assim como, os controles para validação de dados e acionamento de gatilhos na atualização.

3.2 Escopo

Este documento se aplica somente ao módulo de ORM do novo sistema Alphalinc (ALUP), visto que, a arquitetura do módulo foi concebida para atender às necessidades específicas deste projeto. O documento apresenta a arquitetura adotada, os critérios analisados, as restrições identificadas, os recursos de mapeamento de dados, os métodos de acesso existentes, os controles de validação e de atualização, bem como, exemplos de código para referência de utilização pelos desenvolvedores.

3.3 Definições, Acrônimos e Abreviações

- ALUP – Alphalinc Upgrade
- API – Application Programm Interface
- EJB – Enterprise Java Beans
- IDE – Integrated Development Environment
- JDO – Java Data Objects
- JDOQL – JDO Query Language
- JEE – Java Enterprise Edition (J2EE)
- JPA – Java Persistence API
- JPQL – JPA Query Language
- JTA – Java Transaction API
- JVM – Java Virtual Machine

- MDA – Multidimensional Data Access
- NM – Net Manager
- NNM – New Net Manager (Novo Net Manager)
- OODB – Object Oriented Data Base
- ORM – Object Relational Mapping
- POJO – Plain Old Java Object
- RDBMS – Relational Data Base Management System

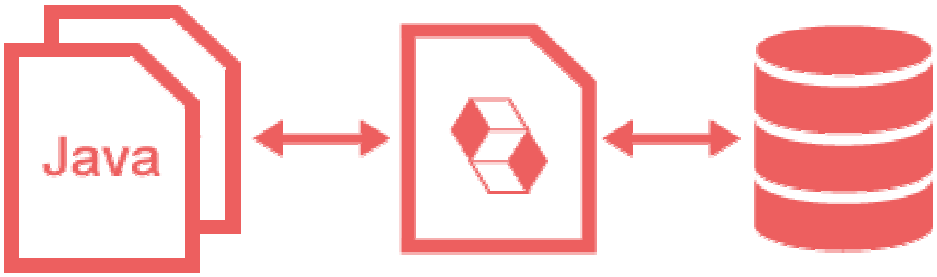
3.4 Referências

- Persistence Specification – JDO x JPA
  https://db.apache.org/jdo/jdo_v_jpa.html
- Wikipedia - List of ORM Software
  http://en.wikipedia.org/wiki/List_of_object-relational_mapping_software
- Open Source Persistence Frameworks in Java
  http://java-source.net/open-source/persistence
- Hibernate ORM
  http://hibernate.org/orm/
- EclipseLink JPA
  https://www.eclipse.org/eclipselink/jpa.php
- DataNucleus Access Platform
  http://www.datanucleus.org/
- Apache EmpireDB - What's wrong with Hibernate and JPA
  http://empire-db.apache.org/empiredb/hibernate.htm
- Simple Java Object Relational Mapping
  http://www.simpleorm.org/
- Apache Cayenne ORM
  https://cayenne.apache.org/

4 Definição da Arquitetura

4.1 Conceito Geral

A integração de fonte de dados em aplicações utilizando a tecnologia Java é um problema altamente complexo e envolve muito mais do que simplesmente ler e escrever um registro no banco de dados. O mapeamento objeto relacional (ORM) é uma técnica que procura solucionar este problema e diminuir a chamada impedância entre o modelo orientado a objetos existente na tecnologia Java e o modelo relacional existente nos principais bancos de dados disponíveis no mercado.



A técnica de ORM foi criada em 2006 para tecnologia Java, existindo duas especificações distintas para solução deste problema: JDO (Java Data Objects) e JPA (Java Persistence API). Apesar da especificação do JDO ser mais completa e contemplar inclusive bancos não relacionais, foi a especificação JPA que conquistou o mercado e possui atualmente o maior número de implementações.

A especificação JPA consiste basicamente de quatro áreas distintas:

- A interface de programação propriamente dita (API);
- A linguagem de pesquisa (JPQL);
- A interface de “Criteria” para pesquisas;
- Os metadados de mapeamento objeto/relacional.

Os metadados de mapeamento objeto/relacional são a ponte que determina como os dados relacionais dos bancos de dados devem ser representados no modelo orientado a objetos, a forma de definição destes metadados pode variar um pouco entre as implementações da biblioteca JPA, contudo existem três formas básicas para declaração dos metadados (nem todas as formas são suportadas por todas as implementações):

- Anotações JPA na classe Java (JPA Annotated POJO);
- Propriedades definidas em arquivos XML;
- Definições dinâmicas geradas de forma programática.

A interface de “Criteria” permite a construção de pesquisas de forma programática sem utilização da linguagem JPQL, este recurso permite a geração de pesquisas dinâmicas em tempo de execução da aplicação sem necessidade de gerar um comando JPQL.

4.2 Objetivos e Restrições

Para definição da arquitetura a ser utilizada no módulo de ORM foram observadas algumas premissas e objetivos específicos do projeto de migração do sistema Alphalinc, segue abaixo os pontos analisados:

- Minimização do esforço para conversão e evolução futura do sistema (a solução deve ser simples e gerar o menor esforço possível de forma a proporcionar maior agilidade na migração e no desenvolvimento);
- Garantia de integridade dos dados e de execução das validações e dos gatilhos estabelecidos (além de integridade dos dados a solução deve garantir que todas as validações e os gatilhos configurados sejam sempre processados na atualização da entidade);
- Máxima transparência ao desenvolvedor (todos os recursos e funcionalidades devem ser implementados com o máximo de transparência ao desenvolvedor, de forma que, o desenvolvedor tenha acesso somente às funções e recursos necessários);
- Otimização de desempenho nas transações junto ao banco relacional (as transações devem ser processadas com um desempenho aproximado do sistema atual em Caché);
- Flexibilidade para alteração da estrutura de dados (via Net Manager deve ser possível a criação e alteração da estrutura de dados com minimização do impacto na aplicação Java).

Com base nestas premissas e objetivos foram identificadas algumas restrições ou mesmo pontos de atenção na definição da arquitetura:

- A utilização de anotações JPA nas classes Java exige um esforço adicional dos desenvolvedores, sendo um ponto crítico para garantia de integridade dos dados e de otimização de desempenho do ORM (mesmo com os recursos de engenharia reversa da IDE para geração desta estrutura);
- A utilização de anotações JPA nas classes Java torna a estrutura de dados mais rígida e estática, dificultando (ou impossibilitando) os recursos para criação e alteração de classes de forma dinâmica no Net Manager;
- A utilização de propriedades definidas em arquivos XML, além de prejudicar o desempenho do ORM, exige o controle de geração e distribuição de mais arquivos do sistema, sendo mais um ponto de vulnerabilidade para falhas;
- O mapeamento de relacionamentos é um ponto extremamente delicado para controle de desempenho de aplicações com utilização do JPA, o desenvolvedor não pode ser responsável pela definição de relacionamentos entre os objetos e da forma de recuperação das informações no banco de dados.

4.3 Critérios de Avaliação

A partir dos objetivos e restrições expostos no tópico anterior foram elencadas duas alternativas de implementação do módulo de ORM quanto à forma de declaração dos metadados de mapeamento objeto/relacional:

- Anotações JPA na classe Java (JPA Annotated POJO);
- Definições dinâmicas geradas de forma programática.

Também foram analisadas as principais implementações JPA existentes no mercado, onde foram selecionadas duas alternativas (ambas são softwares livres sem custo de licenciamento):

- Hibernate – implementação mais utilizada atualmente no mercado (definição dinâmica ainda em fase experimental) – referências RedHat & JBOSS;
- EclipseLink. – implementação de referência JPA pela Oracle (suporta definição dinâmica e alguns bancos de dados não relacionais) – referências Eclipse Foundation & Oracle.

Desta forma a decisão foi tomada com base em duas alternativas finais:

Hibernate – JPA Annotated POJO	**EclipseLink. – Dynamic Entity Definition**
Vantagens (Prós)	
Clareza	**Flexibilidade**
Legibilidade	**Robustez**
Padrão de Mercado	**Menor Esforço**
Facilidade na Programação	**Maior Transparência**
Desvantagens (Contras)	
Maior Esforço	**Pouca Referência de Mercado**
Menor Flexibilidade e Robustez	**Dificuldade na Programação**

4.4 Solução Adotada

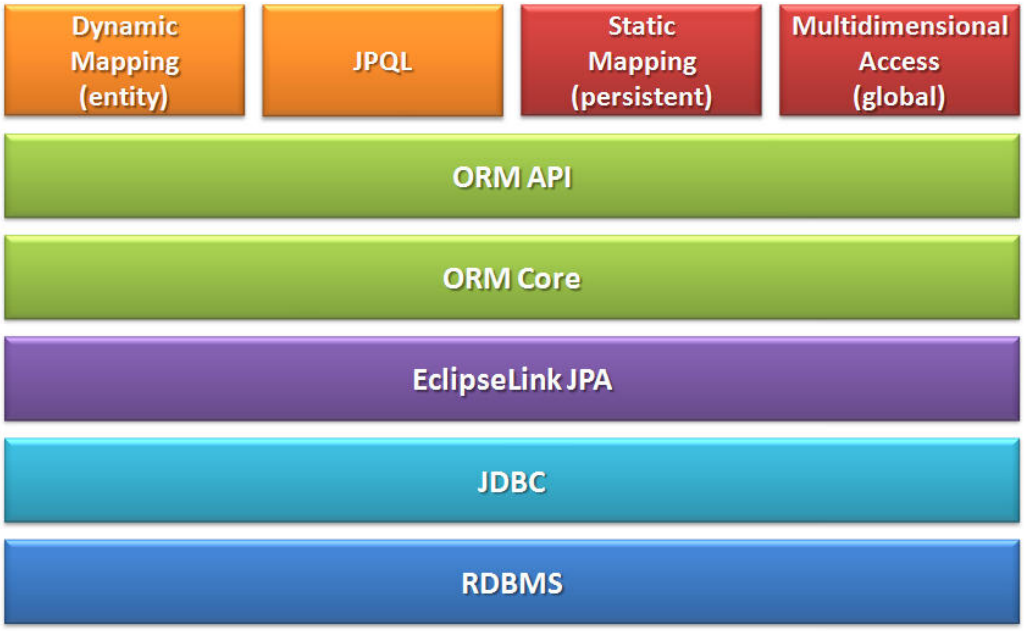
As características de menor esforço para desenvolvimento, maior flexibilidade, maior robustez (com garantia de execução das validações e dos gatilhos configurados) e maior transparência ao desenvolvedor pesaram mais na decisão, sendo assim, a solução

adotada foi baseada no EclipseLink com definição dinâmica dos metadados de mapeamento.

Contudo, a adoção do EclipseLink como biblioteca JPA não atende todas as necessidades do módulo de ORM do sistema Alphalinc, existem algumas lacunas que devem ser suportadas através de camadas e interfaces adicionais:

- Definição automática das classes com base nos metadados;
- Execução das validações e gatilhos configurados via Net Manager;
- Recursos de formatação de dados compatível com a tecnologia M (mumps);
- Recursos de formatação de dados conforme idioma da sessão;
- Suporte para campos calculados e/ou transientes;
- Acesso via mapeamento através de classes persistentes (compatibilidade com legado);
- Acesso via estrutura de dados multidimensional (compatibilidade com legado – globais).

Segue abaixo diagrama geral da arquitetura de camadas e de interfaces do módulo de ORM conforme a solução final adotada:



- As interfaces de acesso em vermelho são para compatibilidade com legado (não devem ser utilizadas em futuras implementações)

5 Mapeamento de Dados

5.1 Classe de Dados

5.2 Tipos de Dados

5.3 Formatação de Dados

5.4 Relacionamentos

5.5 Campos Calculados/Transientes

6 Interfaces de Acesso

6.1 Mapeamento Dinâmico (entity)

6.2 JPQL

6.3 Mapeamento Estático (persistent)

6.4 Acesso Multidimensional (global)

7 Validações & Gatilhos

7.1 Validações de Dados

7.2 Gatilhos de Atualização

7.3 Histórico de Atualização