FACET – Faculdade de Ciências Exatas e Tecnológicas

FASA – Faculdade Santo Agostinho  
Disciplina: Linguagem de Programação IV  
Professor: Petrônio Candido de Lima e Silva  
Equipe: Mariana Mayumi Ferreira Shibuya  
 Jeferson Silveira Santos de Andrade

Tema: JPA - Java Persistense API

**Parte 1: ORM e JPA**

**01)ORM - O que é mapeamento objeto/relacional? Como funciona? Quais os  
principais ORM do mercado?**

É uma técnica de desenvolvimento utilizada para reduzir a impedância da programação orientada aos objetos utilizando bancos de dados relacionais. As tabelas do banco de dados são representadas através de classes e os registros de cada tabela são representados como instancias das classes correspondentes.

Com essa técnica, o programador irá usar uma interface de programação simples que faz todo o trabalho de persistência. Os principais ORM do mercado são o Hiberbate(apache) , Toplink (Oracle), Struts e o Spring.

**02)O que é o padrão JPA? O que ele descreve? De onde surgiu?**

É um padrão da Java para persistência que deve ser implementada por frameworks que queiram seguir o padrão. A JPA define um meio de mapeamento objeto-relacional para objetos Java Simples e Comuns (POJOs), denominados beans de entidade. Diversos frameworks de mapeamento objeto/relacional como o Hibernate implementaram a JPA. Também gerencia o desenvolvimento de entidades do Modelo Relacional usando a plataforma nativa Java SE e Java EE.

**03)Quais as principais implementações do JPA?**

**Hibernate:**

Foi criado por desenvolvedores Java, espalhados ao redor do mundo, e liderado por Gavin King, posteriomente, Jboss Inc. (empresa comprada pela Red Hat). É um frameworks para o mapeamento objeto-relacional escrito na linguagem Java, mas também é disponível em .Net como o nome Nhibernate. Este programa facilita o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação, mediante o uso de arquivos (XML) para estabelecer esta relação. Hibernate é um software livre de código aberto distribuído com a licença LGPL.

O objetivo de Hibernate é diminuir a complexidade entre os programas Java, baseado no modelo orientado objeto, que precisam trabalhar com um bando de dados do modelo relacional (presente na maioria dos SGBDs). Em especial, no desenvolvimento de consultas e atualizações dos dados.

Sua principal característica é a transformação de classes em Java para tabelas de dados (e dos tipos de dados Java para os da SQL). O Hibernate gera as chamadas SQL e libera o desenvolvedor do trabalho manual da conversão dos dados resultante, mantendo o programa portável para quaisquer bancos de dados SQL, porém causando um pequeno aumento no tempo de execução.

**Toplink:**

Toplink foi desenvolvido originalmente pelos povos do objeto dentroSmalltalk nos 1990s. O “ALTO” em Toplink é um acrônimo para os povos do objeto, e o nome capitalized originalmente como “Toplink”. Em 1996-1998 uma versão de Java do produto foi adicionada à linha de produto, nomeada Toplink para Java. Após a aquisição comum dos povos do objeto em abril 2000 perto Sistemas de BEA e Webgain, A linha de produto de Toplink transformou-se a propriedade de Webgain.

Em 2002, Toplink foi adquirido perto Oracle Corporatione continua a ser tornado como um membro do Middleware da fusão de Oraclefamília dos produtos.

Em 2006, Oracle doou o código de fonte do produto de Toplink e dos recursos do desenvolvimento à fonte aberta Sun Microsystems java.net Glassfish projeto. Este projeto foi nomeado fundamentos de TopLink e era a execução da referência de Java EE EJB 3.0 JPA.

Em 2007, Oracle doou o código de fonte do produto de TopLink e dos recursos do desenvolvimento à fonte aberta Fundação do eclipse EclipseLink projeto.

Em março 2008, a fundação do eclipse, anunciadaesse Sun Microsystems selecionou o projeto de EclipseLink como a execução da referência para o JPA 2.0, padrão de JSR 317 para suceder fundamentos de TopLink.

O TopLink é um dos possíveis provedores de persistência para a JPA. Ele provê alta performance e escalabilidade, produtividade para o desenvolvedor e flexibilidade. O TopLink trabalha em conjunto com qualquer banco de dados, incluindo banco de dados não relacionais, e com qualquer servidor de aplicação.

O TopLink provê também recursos como:

* Um rico framework que provê opções de fácil uso para formular queries dinâmicas e sofisticadas. Onde os desenvolvedores podem definir queries usando qualquer expressão, como EJB QL, SQL e Stored Procedures.
* Um framework de transação que provê uma transação a nível de objetos. O framework de transação suporta gerenciamento direto das transações, e também gerenciamento externo das transações através de JTA/JTS.
* Um sistema avançado de cache que melhora a performance do sistema, ao não requerer que os dados mais recentemente ou mais frequentemente acessados sejam recuperados do banco de dados, mas sim da memória do servidor.

**EclipseLink**

Os líderes da indústria da Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft WebGain se juntaram e formaram a primeira eclipse.org Conselho de Comissários Desportivos, em novembro de 2001. Até o final de 2003, este consórcio inicial tinha crescido para mais de 80 membros.

Em 2 de fevereiro de 2004 o Conselho de Regentes Eclipse anunciou a reorganização Eclipse em uma organização sem fins lucrativos corporação. Originalmente um consórcio que se formou quando a IBM lançou a Plataforma Eclipse Open Source, Eclipse tornou-se um órgão independente que irá conduzir a evolução da plataforma de vantagem para os prestadores de ofertas de desenvolvimento de software e usuários finais. Todo o código fonte e tecnologia fornecidos e desenvolvido por esta comunidade em rápido crescimento é disponibilizada a título gratuito através da Eclipse Public License.

Eclipse é uma comunidade de fonte aberta, cujos projetos são focados em construir uma plataforma de desenvolvimento aberta extensível composto por quadros, ferramentas e tempos de execução para a construção, implantação e gerenciamento de software em todo o ciclo de vida. A Eclipse Foundation é uma organização sem fins lucrativos, apoiada sobre as sociedades membros que hospeda os projetos Eclipse e ajuda a cultivar tanto a comunidade open source e um ecossistema de produtos e serviços complementares.

O projeto Eclipse foi originalmente criado pela IBM em Novembro de 2001 e apoiado por um consórcio de fornecedores de software. A Eclipse Foundation foi criada em janeiro de 2004 como um independente sem fins lucrativos para a empresa para atuar como o mordomo da comunidade Eclipse. A organização independente sem fins lucrativos para a empresa foi criada para permitir um fornecedor neutro e comunidade, aberta e transparente a ser estabelecida em torno do Eclipse. Hoje, a comunidade Eclipse é composta por indivíduos e organizações a partir de uma seção transversal da indústria de software.

A Eclipse Foundation é financiada por contribuições anuais dos nossos membros e regido por um Conselho de Administração. Estratégico desenvolvedores e consumidores Estratégica lugares cativos nesta Câmara, assim como representantes eleitos pelo add-in Provedores e committers Open Source. A Fundação emprega um em tempo integral equipe de profissionais para prestação de serviços à comunidade, mas não empregar os desenvolvedores de código aberto, committers chamado, o que realmente trabalhar nos projetos do Eclipse. Committers Eclipse são normalmente empregadas por organizações ou é desenvolvedores independentes, que doam seu tempo para trabalhar em um projeto open source.

**OpenJPA**

**Kodo**, a Objetos dos dados de Java a execução, foi desenvolvida originalmente por SolarMetric, Inc em 2001. Sistemas de BEA SolarMetric adquirido em 2005, onde Kodo foi expandido para ser uma execução de ambos os JDO (JSR 12) e JPA (JSR 220) especificações. Em 2006, BEA doou uma parte grande do código de fonte de Kodo à Fundação do software de Apache sob o nome **OpenJPA**. OpenJPA será o motor do persistence do núcleo de Usuário de BEA Weblogic, IBM WebSphere, e Geronimo Usuário da aplicação. Em maio 2007, em OpenJPA graduado do incubador a um projeto top-level e a um Sun também passado Jogo da compatibilidade da tecnologia compliant com o Persistence API de Java.

OpenJPA Apache é um projeto de persistência em Java da Apache Software Foundation , que pode ser utilizado como um stand-alone POJO camada de persistência ou integrado em qualquer contêiner Java EE compliant e muitos outros quadros leves, como Tomcat e Primavera.

As versões 1.x (1.2.2 é a última) são uma produção pronta, rico em recursos compatíveis implementação da Java Persistence API (JPA) 1.0 parte da JSR-220 Enterprise Beans Java 3.0 da especificação, que passam a APP dom. Kit de Compatibilidade 1.0b Tecnologia.

As versões 2.x (2.1.0 é a última) são uma produção compatível, a execução do pronto -317 Java Persistence JSR 2,0 especificação, o que é compatível com a especificação do JPA 1,0 e passa o Sol JPA 2.0 Technology Compatibility Kit.

A próxima versão 2.2 é baseado fora do fluxo de liberação 2.1.Xe irá conter algumas funcionalidades adicionais (TBD).

**04)Como se configura o JPA? (persistence.xml)**

*<?xml version="1.0" encoding="UTF-8"?>*

*<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence">*

*<persistence-unit name="manager1" transaction-type="JTA">*

*<provider>org.hibernate.ejb.HibernatePersistence</provider>*

*<jta-data-source>java:/DefaultDS</jta-data-source>*

*<mapping-file>ormap.xml</mapping-file>*

*<jar-file>MyApp.jar</jar-file>*

*<class>org.acme.Employee</class>*

*<class>org.acme.Person</class>*

*<class>org.acme.Address</class>*

*<shared-cache-mode>ENABLE\_SELECTIVE</shared-cache-mode>*

*<validation-mode>CALLBACK</validation-mode>*

*<properties>*

*<property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />*

*<property name="hibernate.connection.url" value="jdbc:mysql://localhost:3306/database" />*

*<property name="hibernate.connection.driver\_class" value="com.mysql.jdbc.Driver" />*

*<property name="hibernate.connection.password" value="" />*

*<property name="hibernate.connection.username" value="root" />*

*<property name="hibernate.hbm2ddl.auto" value="create-drop" />*

*</properties>*

*</persistence-unit>*

*</persistence>*

Para utilizar outros Bancos de Dados, basta alterar a url de conexão e o driver utilizado, por exemplo, no caso do Oracle:

*<property name="hibernate.dialect" value="org.hibernate.dialect.OracleDialect" />*

*<property name="hibernate.connection.url" value="jdbc:oracle:thin:@localhost:1521:DB" />*

*<property name="hibernate.connection.driver\_class" value="oracle.jdbc.OracleDriver" />*

Para o HSQLDB:

*<property name="hibernate.dialect" value="org.hibernate.dialect.HSQLDialect"/>*

*<property name="hibernate.connection.url" value="jdbc:hsqldb:." />*

*<property name="hibernate.connection.driver\_class" value="org.hsqldb.jdbcDriver" />*

Em password e username você deve utilizar os dados para acessar o seu BD.

**Name:** (Atributo) Todo gerente de entidade deve ter um nome.

**Transaction-type:** (Atributo) tipo de transação usado. Ou JTA ou RESOURCE\_LOCAL (padrão em um ambiente JTA JavaEE e RESOURCE\_LOCAL em um ambiente de JavaSE). Quando um datasource JTA é usado, o padrão é JTA, se não datasource JTA é usado, RESOURCE\_LOCAL é usado.

**Provider:** É um nome de classe totalmente qualificado do provedor de persistência EJB. Isto é necessário quando você estiver usando implementações de vários fornecedores de persistência EJB.

**Jta-data-source, non-jta-data-source:** É o JNDI do local onde o javax.sql.DataSource está situado. Ao ser executado sem um DataSource JNDI disponível, você deve especificar conexões JDBC com as propriedades especificas do Hibernate.

**Mapping-file:** O elemento de classe especifica uns arquivos de mapeamento EJB3 XML compatível que irá mapear. O arquivo deve estar no classpath. De acordo com a especificação do EJB3, Hibernate EntityManager irá tentar carregar o arquivo de mapeamento localizado no arquivo jar em Meta\_INF/orm.xml. É claro que qualquer arquivo de mapeamento explícito serão carregados também. Por uma questão de fato, você pode fornecer qualquer arquivo XML no ie elemento de mapeamento de arquivo ou arquivos hbm ou descritor de implantação EJB3.

**Jar-file:** Os elementos jar especifica uma jar para analisar. Todas as classes devidademente anotado e todos os pacotes de parte dos arquivos hbm.xml deste arquivo jar será adicionada à configuração da unidade de persistência. Pode-se alternativamente apontar para um diretório (Isto é especialmente útil quando em seu ambiente de teste, o arquivo persistence.xml não está sob o mesmo diretório raiz o que o seu modelo de domínio).

**Class:** O elemento de classe especifica um nome de classe totalmente qualificado que irá mapear. Por padrão todas as classes devidamente anotadas e todos os arquivos hbm.xml encontrada dentro do arquivo são adicionados à configuração da unidade de persistência. Você pode adicionar alguma entidade externa através do elemento de classe embora. Como uma extensão da especificação, você pode adicionar um nome de pacote na <class> elemento (por exemplo <class>org.hibernate.eg</class> ).

**Shared-cache-mode:** Por padrão, as necessidades são eleitas para cachê de segundo nível se anotado com @cacheale. Você pode, no entanto:

* ALL: Forçar cachê para todas as entidades.
* None: Desativa cachê para todas as entidades.
* Enable\_selective: Habilitar o cachê quando explicitamente marcada.
* Disable\_Selective: Habilita o cachê a menos que explicitamente marcado como @Cacheable(false) (não recomendado).

**Validation-mode:** Por padrão, Bean Validation (e Hibernate Validator) é ativado. Quando uma entidade é criada, atualizado (e, opcionalmente, excluída), é validada antes de ser enviado para o banco de dados. O esquema do banco de dados gerado pelo Hibernate também reflete as restrições declaradas na entidade.

Pode-se fazer um ajuste fino que, se necessário:

* AUTO: se validação Bean está presente no classpath, CALLBACK e DDL são ativados.
* CALLBACK: entidades são validadas na criação, atualização e exclusão. Se nenhum provedor de validação Bean está presente, é levantada uma exceção em tempo de inicialização.
* DDL: (no standart, ver abaixo): são entidades esquemas de banco de dados e são validados na atualização, criação e exclusão. Se nenhum provedor de validação Bem está presente, é levantada uma exceção em tempo de inicialização.
* NONE: Bean Validatin não e usado para todos.

Para usa adicione uma propriedade regular.

*<property name="javax.persistence.validation.mode">*

*ddl*

*</property>*

Com essa abordagem, pode-se misturar os modos ddl e retorno:

*<property name="javax.persistence.validation.mode">*

*ddl, callback*

*</property>*

**Properties:** O Elemento de propriedade é usado para especificar as propriedades específicas do fornecedor. Isto é onde você irá definir as configurações específicas do seu Hibernate. E onde você terá que especificar informações de conexão JDBC também.