Framework JPA

JPA é um framework leve, baseado em **POJOS** (Plain Old Java Objects) para persistir objetos Java. A Java Persistence API, diferente do que muitos imaginam, não é apenas um framework para Mapeamento Objeto-Relacional (ORM - Object-Relational Mapping), ela também oferece diversas funcionalidades essenciais em qualquer aplicação corporativa.

Atualmente temos que praticamente todas as aplicações de grande porte utilizam JPA para persistir objetos Java. JPA provê diversas funcionalidades para os programadores, como será mais detalhadamente visto nas próximas seções. Inicialmente será visto a história por trás da JPA, a qual passou por algumas versões até chegar na sua versão atual.

História da Especificação

Após diversos anos de reclamações sobre a complexidade na construção de aplicações com Java, a especificação Java EE 5 teve como principal objetivo a facilidade para desenvolver aplicações JEE 5. O EJB 3 foi o grande percursor para essa mudança fazendo os Enterprise JavaBeans mais fáceis e mais produtivos de usar.

No caso dos Session Beans e Message-Driven Beans, as soluções para questões de usabilidade foram alcançadas simplesmente removendo alguns dos mais onerosos requisitos de implementação e permitindo que os componentes sejam como Plain Java Objects ou POJOS.

Já os Entity Beans eram um problema muito mais sério. A solução foi começar do zero. Deixou-se os Entity Beans sozinhos e introduziu-se um novo modelo de persistência. A versão atual da JPA nasceu através das necessidades dos profissionais da área e das soluções proprietárias que já existiam para resolver os problemas com persistência. Com a ajuda dos desenvolvedores e de profissionais experientes que criaram outras ferramentas de persistência, chegou a uma versão muito melhor que é a que os desenvolvedores Java conhecem atualmente.

Dessa forma os líderes das soluções de mapeamento objetos-relacionais deram um passo adiante e padronizaram também os seus produtos. Hibernate e TopLink foram os primeiros a firmar com os fornecedores EJB.

O resultado final da especificação EJB finalizou com três documentos separados, sendo que o terceiro era o Java Persistence API. Essa especificação descrevia o modelo de persistência em ambos os ambientes Java SE e Java EE.

JPA 2.0

No momento em que a primeira versão do JPA foi iniciada, outros modelos de persistência ORM já haviam evoluído. Mesmo assim muitas características foram adicionadas nesta versão e outras foram deixadas para uma próxima versão.

A versão JPA 2.0 incluiu um grande número de características que não estavam na primeira versão, especialmente as mais requisitadas pelos usuários, entre elas a capacidade adicional de mapeamento, expansões para a Java Persistence Query Language (JPQL), a API Criteria para criação de consultas dinâmicas, entre outras características.

JPA 2.1

Main features included were:

- Converters - allowing custom code conversions between database and object types.
- Criteria Update/Delete - allows bulk updates and deletes through the Criteria API.
- Entity Graphs - allow partial or specified fetching or merging of objects.
- JPQL/Criteria enhancements - arithmetic sub-queries, generic database functions, join ON clause, TREAT option.
- Schema Generation
- Stored Procedures - allows queries to be defined for database stored procedures.

Vendors supporting JPA 2.1:

- DataNucleus
- EclipseLink
- Hibernate

Entre as principais inclusões na JPA destacam-se:

- **POJOS Persistentes:** Talvez o aspecto mais importante da JPA seja o fato que os objetos são POJOs (Plain Old Java Object ou Velho e Simples Objeto Java), significando que os objetos possuem design simples que não dependem da herança de interfaces ou classes de frameworks externos. Qualquer objeto com um construtor default pode ser feito persistente sem nenhuma alteração numa linha de código. Mapeamento Objeto-Relacional com JPA é inteiramente dirigido a metadados. Isto pode ser feito através de anotações no código ou através de um XML definido externamente.

- **Consultas em Objetos:** As consultas podem ser realizadas através da Java Persistence Query Language (JPQL), uma linguagem de consulta que é derivada do EJB QL e transformada depois para SQL. As consultas usam um esquema abstraído que é baseado no modelo de entidade como oposto às colunas na qual a entidade é armazenada.
- **Configurações simples:** Existe um grande número de características de persistência que a especificação oferece, todas são configuráveis através de anotações, XML ou uma combinação das duas. Anotações são simples de usar, convenientes para escrever e fácil de ler. Além disso, JPA oferece diversos valores defaults, portanto para já sair usando JPA é simples, bastando algumas anotações.
- **Integração e Teste:** Atualmente as aplicações normalmente rodam num Servidor de aplicação, sendo um padrão do mercado hoje. Testes em servidores de aplicação são um grande desafio e normalmente impraticáveis. Efetuar teste de unidade e teste caixa branca em servidores de aplicação não é uma tarefa tão trivial. Porém, isto é resolvido com uma API que trabalha fora do servidor de aplicação. Isto permite que a JPA possa ser utilizada sem a existência de um servidor de aplicação. Dessa forma, testes unitários podem ser executados mais facilmente.

Conclusão

Destacou-se o que é JPA, como JPA evoluiu ao longo do tempo e o que influenciou as características presentes na sua versão atual, considerada uma grande evolução na plataforma Java. Além disso, tivemos a oportunidade de verificar quais são as principais inclusões na atual versão da JPA e como eles podem impulsionar o desenvolvimento de aplicações corporativas que exigem uma manutenção mais simples, fácil e rápida além de confiabilidade e testes de unidades, indispensável em qualquer aplicação corporativa hoje.

Bibliografia

- http://www.devmedia.com.br/introducao-a-jpa-java-persistence-api/28173
- https://en.wikipedia.org/wiki/Java_Persistence_API