Armazenando Dados em Aplicações Java

Parte 2 de 3: Apresentando as opções

Hua Lin Chang Costa, hualin@cos.ufrj.br, COPPE/UFRJ.
Leonardo Gresta Paulino Murta, leomurta@ic.uff.br, IC/UFF.
Vanessa Braganholo, braganholo@dcc.ufrj.br, DCC/UFRJ.

Quais são os problemas e as opções disponíveis para armazenamento de dados em aplicações Java de verdade? Este artigo é o segundo de uma trilogia, e responde a parte "quais são as opções disponíveis" dessa pergunta.

Quando estamos em um ambiente acadêmico e precisamos fazer um programa para alguma disciplina ou trabalho final de curso, o armazenamento dos dados é de pouca importância, e soluções como serialização normalmente são suficientes. Contudo, depois de formados precisaremos lidar com sistemas reais, em ambientes reais, e a solução anterior deixa de ser satisfatória. Este artigo é o segundo de uma trilogia, e apresenta quais são as tecnologias disponíveis para tratar o problema de armazenamento de dados em Java, assim como quais são as características principais dessas tecnologias.

No primeiro artigo da trilogia analisamos um caso real: o Ambiente Odyssey, que possui mais de 10 anos de história, com novas funcionalidades implementadas a cada ano por alunos do grupo de Reutilização da COPPE/UFRJ. Naquele artigo, através dos resultados de uma pesquisa, chegamos aos seguintes requisitos para um mecanismo de persistência: prover compatibilidade entre diferentes versões do sistema, ser otimizado em termos de desempenho, ter confiabilidade, ser de fácil manutenção e de evoluir e não ser intrusivo.

Esse artigo leva em consideração esses requisitos para selecionar as principais especificações e *frameworks* de armazenamento disponíveis para a linguagem Java. Para isso, iniciamos com uma breve discussão que contrasta especificação de implementação. Em seguida apresentamos as duas principais especificações para armazenamento em Java: JDO e JPA. Finalmente, descrevemos alguns dos *frameworks* disponíveis para esse fim.

Especificação x Implementação

É importante ressaltar a diferença entre especificação e implementação. Especificações são documentos que definem padrões para determinados tipos de processos. Estas especificações, definidas pelo processo da comunidade Java (JCP), podem ou não ser implementadas e adicionados à plataforma Java no futuro. Já as implementações discutidas neste artigo são *frameworks* para armazenamento. Esses frameworks podem ou não ser implementações de uma especificação. Um exemplo de não aderência a especificações é o Castor, que será visto mais adiante.

Especificações

***Java Persistence API* (JPA)** é a especificação que define um padrão de persistência para permitir que desenvolvedores Java gerenciem dados que são armazenados de forma relacional. A especificação JPA está inclusa no documento 'JSR: 220 Enterprise JavaBeans 3.0', que é a especificação da arquitetura Enterprise JavaBeans para desenvolvimento de aplicações baseadas em componentes. A especificação descreve os requisitos para o suporte à Enterprise JavaBeans 3.0, mas permite também o seu uso de forma isolada através da plataforma Java SE.

A JPA é um conjunto de interfaces que descrevem como gerenciar o conteúdo de banco de dados relacionais. Para isso, ela descreve uma linguagem de consulta e um conjunto de metadados para configuração do mapeamento objeto relacional. A linguagem de consulta JPQL (*Java Persistence*

Query Language) é muito similar à linguagem SQL, mas atua sobre os objetos e não diretamente sobre as tabelas de um banco de dados relacional.

A construção do padrão de persistência JPA teve a contribuição de vários especialistas da comunidade Java. Assim, a especificação incorpora contribuições de várias soluções populares do mercado como Hibernate e TopLink.

***Java Data Objects* (JDO)** é uma especificação que define uma API para persistência transparente de objetos. Atualmente ela se encontra na versão 2.0 e é descrita no documento 'JSR: 243 – Java Data Objects 2.0'.

Diferentemente da especificação JPA, a especificação JDO, além de definir o mapeamento objeto-relacional como possível mecanismo de persistência, é fundamentalmente um padrão de persistência transparente. Assim, a especificação JDO não restringe a tecnologia do meio de armazenamento, ficando a critério dos implementadores a possibilidade de oferecer suporte não só a banco de dados relacionais como também a outras formas de armazenamento, incluindo banco de dados orientados a objetos.

Frameworks

XStream é uma ferramenta para mapeamento entre objetos Java e documentos XML, permitindo assim que objetos possam ser serializados em documentos XML.

XStream foi projetada para que se tenha o menor trabalho possível em termos de configuração e mapeamento entre o objeto Java e a estrutura do documento XML onde os dados serão persistidos. Através de um conversor padrão a ferramenta faz uso de reflexão para descobrir dinamicamente os atributos de uma classe Java e seus respectivos valores. Assim, o mapeamento não precisa ser feito manualmente. Essa falta de necessidade de configuração pode ser atrativa em termos de facilidade e simplicidade, mas acaba tornando o mecanismo pouco flexível e pouco robusto quando se deseja mapeamentos complexos.

Castor XML é um framework que permite a representação de dados de documentos XML em objetos. Assim, o modelo de objetos Java é capaz de manipular e gerenciar as informações contidas em um documento XML. O framework Castor provê a serialização e desserialização entre objetos Java e documentos XML.

Assim como a ferramenta XStream, o framework Castor permite um mapeamento automático entre objetos e documentos XML através de reflexão. Mas para que se obtenha maior controle sobre o mapeamento o mais indicado é a realização manual desse mapeamento a partir de funcionalidades oferecidas pelo próprio framework.

Torque é um framework de persistência para Java, onde os objetos são armazenados através do mapeamento objeto-relacional. O projeto Torque é mantido pela Apache *Software Foundation* e distribuído pela licença Apache 2.

Diferente de outras ferramentas que também fazem uso do mapeamento objeto-relacional, o Torque não faz uso de reflexão para acessar as propriedades das classes Java e então gerar um banco de dados que atenda um mapeamento. No Torque as classes a serem persistidas são geradas a partir de um esquema XML que descreve, além do mapeamento objeto-relacional, o projeto do banco de dados. Assim, as classes Java a serem persistidas não precisam ser programadas visto que elas são geradas a partir do que foi modelado no esquema XML.

O principal objetivo do Torque é o uso de Java para manipulação de um banco de dados. Isso facilita as operações com informações visto que a linguagem de consulta própria do banco de dados é substituída por métodos Java. Assim, o modelo de persistência acaba perdendo um pouco de sua transparência, visto que nesse caso o modelo relacional tem reflexo direto no modelo de classes e não o contrário.

Desse modo, como o projeto Torque foca diretamente na criação de um conjunto de classes a partir de um modelo relacional, o esforço em realizar esse mapeamento quando já se tem um modelo de classes bem definido pode acabar gerando um esforço consideravelmente maior frente a abordagens mais transparentes.

Hibernate é um framework de persistência de código aberto e distribuído pela licença LGPL. Ele provê a persistência de objetos Java através do mapeamento objeto-relacional. Além de prover suporte para o mapeamento objeto-relacional, também provê linguagens de consulta e métodos de gerenciamento de transação que facilitam o processo de desenvolvimento e tornam a persistência o mais transparente possível.

Devido à sua robustez e maturidade, o Hibernate é atualmente um dos frameworks de persistência mais usados no mercado. Muitas de suas funcionalidades foram incorporadas a padrões de persistência em Java e suas versões mais recentes (a partir da versão 3.0) são implementações da especificação JPA.

JPOX é um framework de persistência heterogêneo para Java. A heterogeneidade da solução está no fato dela oferecer suporte a diferentes especificações: JDO1, JDO2 e JPA. Um dos principais objetivos do JPOX é prover um modelo de persistência transparente e flexível através do suporte a diferentes APIs, bancos de dados relacionais, formas de armazenamento e linguagens de consulta. Atualmente o JPOX (versão 1.2) é a implementação de referência da especificação JDO 2.0.

Através do JPOX é possível seguir as diferentes especificações JDO ou JPA. Independente da API escolhida, o modelo de persistência é transparente e pouco intrusivo. Apesar da especificação JDO possibilitar o uso de diferentes formas de armazenamento, o uso de mecanismos diferentes de banco de dados relacionais ainda é muito insipiente. Assim o mecanismo de armazenamento, independente da escolha de especificação, é provido por um banco de dados relacional.

TopLink é um framework de persistência desenvolvido pela Oracle que permite o mapeamento de objetos para bancos de dados relacionais ou documentos XML através de seus diferentes módulos.

Baseado no TopLink, foi desenvolvido o TopLink Essentials, uma versão *open-source* do framework de persistência da Oracle e atual implementação de referência da especificação JPA. O framework Top Link Essentials não oferece todas as funcionalidades da versão original e se limita ao modelo de persistência definido pela especificação JPA. Assim o mecanismo de armazenamento se dá através de bancos de dados relacionais.

Análise das opções

A tabela a seguir relaciona as diferentes características das tecnologias. Ferramentas como XStream e Castor são soluções alternativas ao mapeamento objeto-relacional e fazem uso de documentos XML para o armazenamento de informações. As vantagens no uso dessas soluções residem na flexibilidade do formato XML. A desvantagem dessa forma de armazenamento está na dificuldade de mapear objetos Java para documentos XML. Essa tarefa se torna cada vez mais custosa quando se aumenta a complexidade do modelo de classes.

As outras tecnologias oferecem solução de persistência através do mapeamento objeto-relacional. São elas: Torque, Hibernate, JPOX e TopLink Essentials. Ao contrário das ferramentas XStream e Castor, essas outras soluções são mais robustas e se sustentam sobre especificações de modelos de persistência padrões em Java, com exceção do framework Torque que apresenta um padrão de persistência próprio e não segue nenhuma especificação. O Hibernate e TopLink Essentials implementam a especificação JPA, enquanto o framework JPOX oferece suporte tanto à especificação JPA quanto à JDO.

Ferramenta	Forma de armazenamento	Especificação	Licença
XStream	Documentos XML	-	BSD
Castor	Documentos XML	-	Apache 2.0
Torque	Banco de dados relacional	-	Apache 2.0
Hibernate	Banco de dados relacional	JPA	LGPL
JPOX	Variável	JPA & JDO	Apache 2.0
TopLink Essentials	Banco de dados relacional	JPA	CDDL

As tecnologias que utilizam bancos de dados relacionais como forma de armazenamento se distinguem basicamente pela especificação que implementam. A tabela a seguir relaciona as principais características entre as especificações JDO e JPA.

Especificação	Padrão	Modelo de metadados	Mecanismo de armazenamento
JPA	Mapeamento objeto relacional	XML ou anotações	Banco de dados relacional
JDO	Persistência de dados transparente	XML ou anotações	Transparente em relação ao mecanismo. Depende da implementação.

É importante destacar que ambas as especificações permitem o uso de anotações ou de um descritor XML para que o usuário configure o mapeamento objeto-relacional. Anotações são colocadas nas próprias classes Java através do uso de @. O código abaixo ilustra o uso de anotações.

```
@Entity
public class Book extends Product {
  @Basic
  private String author = "";

  @Basic
  private String isbn = "";

  @OneToMany(cascade=CascadeType.ALL,
                    mappedBy="book")
  private Collection<Review> reviews = null;
```

O descritor XML, por sua vez, é um arquivo XML à parte que especifica o mapeamento. Um exemplo de arquivo descritor é mostrado abaixo.

```
<entity class="br.ufrj.dcc.hualin.Study.model.Book">
    <attributes>
        <basic name="author"/>
        <basic name="isbn"/>
        <one-to-many name="reviews" mapped-by="book">
            <cascade>
                  <cascade-all/>
            </cascade>
        </one-to-many>
    </attributes>
</entity>
```

Concluindo

A partir desse artigo, pudemos ter uma visão geral das principais opções de armazenamento de dados em Java. No próximo artigo da trilogia exemplificaremos como algumas dessas opções funcionam, viabilizando a escolha da opção mais adequada para o seu problema específico.

Recursos

Leia mais sobre a especificação JDO: http://www.jcp.org/en/jsr/detail?id=243
Leia mais sobre a especificação JPA: http://jcp.org/en/jsr/detail?id=220
Leia mais sobre o *framework* XStream: http://xstream.codehaus.org/index.html
Leia mais sobre o *framework* Castor: http://www.castor.org/xml-framework.html
Leia mais sobre o *framework* Torque: http://db.apache.org/torque/documentation/index.html
Leia mais sobre o *framework* Hibernate: http://www.hibernate.org/
Leia mais sobre o *framework* JPOX: http://www.jpox.org/docs/index.html
Leia mais sobre o *framework* TopLink Essentials:
http://www.oracle.com/technology/products/ias/toplink/doc/1013/main/_html/toc.htm

Sobre os autores

Hua Lin Costa é aluno de mestrado do Programa de Engenharia de Sistemas e Computação (COPPE/UFRJ) na linha de pesquisa de Engenharia de Software. Possui graduação em Ciência da Computação (2008) pela UFRJ. Atualmente realiza trabalhos acadêmicos junto ao Grupo de Reutilização de Software da COPPE como colaborador no desenvolvimento do ambiente Odyssey e do Projeto Brechó.

Leonardo Murta é sócio da SBC e professor de Engenharia de Software do Instituto de Computação da Universidade Federal Fluminense. Possui graduação em Informática (1999) pela UFRJ e mestrado (2002) e doutorado (2006) em Engenharia de Sistemas e Computação, também pela UFRJ. Suas principais áreas de interesse são gerência de configuração, reutilização e arquiteturas e software. Mais informações podem ser obtidas em http://www.ic.uff.br/~leomurta.

Vanessa Braganholo é sócia da SBC e professora do Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal do Rio de Janeiro. Possui graduação em Ciência da Computação (1998) pela Universidade Federal do Rio Grande do Sul e doutorado (2004) em Ciência da Computação também pela UFRGS. Tem atuado em diversos projetos de pesquisa ligados a dados semi-estruturados. Sua principal área de atuação é bancos de dados. Mais informações podem ser obtidas em http://www.dcc.ufrj.br/~braganholo.