PERSISTE: Framework para persistência de dados isolada à regra de negócios.

Lucas Rubian Schatz[1], Eloir Viecelli[1], Vander Vigolo[1]

[1] Universidade Do Oeste de Santa Catarina (UNOESC) – Joaçaba – SC – Brasil

{lucasschatz, eloirviecelli}@gmail.com, vandervigolo@yahoo.com.br

Abstract. *The growing need to develop software in a shorter time, gives rise to new tools and frameworks to support developers. The major task for the development of systems is the persistence layer of the data. Observing this lack, this paper proposes to develop a framework to improve the development of the layer for persistence objects in a relational database using Java, and to make easier the persistence layer for programmers and students from computer science, in understanding the concepts of Object Relational Mapping. With the purpose of evaluating the subjective aspects of usability by developing a software prototype, the results shows that 90,48% of interviewed intent to use it again, and 80,96% thinks that it is extremely simple to use the framework..*

Keywords: *Framework, Persistence, Object-Relational Mapping.*

Resumo. *A crescente necessidade de desenvolver softwares em um menor espaço de tempo, faz com que surjam novas ferramentas e frameworks de apoio aos desenvolvedores. Umas das tarefas importantes para o desenvolvimento de sistemas é o desenvolvimento da camada de persistência dos dados. Com relação a esta lacuna a proposta deste trabalho é desenvolver um framework que agilize o desenvolvimento da camada de persistência de objetos em um banco de dados relacional utilizando Java e facilite aos programadores e alunos dos cursos de informática, na compreensão dos conceitos sobre Mapeamento Objeto Relacional. Com o intuito de avaliar a usabilidade do Persiste, foi elaborado um questionário e a apuração dos resultados mostra que 90,48% dos entrevistados pretendem utilizar o framework novamente, e 80,96% acham extremamente simples a sua utilização.*

Palavras-chave*: Framework, Persistência, Mapeamento Objeto Relacional.*

1. Introdução

Atualmente, ao iniciar um projeto, não há muito interesse em se preocupar com a persistência dos dados, mas sim, nas regras de negócio que são o foco principal do sistema. Muitos programadores que não utilizam *frameworks* de persistência gastam um tempo considerável preparando toda a persistência dos dados, desviando seus esforços a uma atividade fundamental, contudo secundária. Segundo REGO (2006), "a persistência

dos dados consiste no armazenamento confiável e coerente das informações em um sistema de armazenamento de dados".

Segundo Prado (2009), "o código relacionado à persistência dos dados, pode ser uma das partes mais 'tediosas' em um aplicação", por ser uma parte extremamente repetitiva, com o mesmo padrão de programação para todas as entidades, segundo o modelo de projeto MVC (*Model View Cotroller*).

Visando otimizar o tempo do programador, o projeto proposto busca automatizar a persistências dos dados, através de classes com atributos que representam as tabelas e colunas de um banco de dados, ficando a cargo do framework Persiste a inserção, atualização, leitura e exclusão dos registros em um banco de dados relacional, não exigindo do programador o uso intensificado dos comandos select, insert, update e delete do padrão SQL. Quando houver a requisição para persistir ou gravar um objeto, ele será desmaterializado e inserido ou alterado no banco de dados, e quando for necessário recuperar um objeto do banco, ele será materializado e retornado para a aplicação. Segundo Rosa (2003) a "materialização é o ato de transformar uma representação de dados não-orientada ao objeto (registros), existente em um armazenamento persistente em objetos. A desmaterialização é a atividade oposta que também é conhecida como passivação".

2. Desenvolvimento

Macoratti (2003) diz que o projeto é dividido em três partes: a camada de apresentação, onde o usuário interage; a camada de controle, onde residem as regras de negócio e; a camada do modelo, que representam as entidades correspondentes às classes juntamente com a persistência, que por sua vez, é influenciada diretamente pela entidade.

As classes de persistência são dependentes da entidade, se houver qualquer alteração da mesma, e for necessário persistir a informação adicional, será necessário alterar as operações básicas de leitura, escrita, atualização e exclusão. "As operações Criar, Ler, Atualizar e Excluir (CRUD) são as operações de banco de dados mais básicas, mas também são as mais cruciais. As operações CRUD são geralmente realizadas usando Linguagem de Consulta Estruturada (SQL) em sistemas de bancos de dados relacionais" [Lewis 2008]. Este tipo de dependência acaba por prejudicar o trabalho do programador, pois ele deve controlar as diferenças entre a entidade e o CRUD manualmente, como por exemplo a inclusão de um novo campo.

Com a popularização de metodologias de programação orientadas a objetos, houve uma mudança na forma como o programador realiza a persistência dos dados em seus projetos. Contudo, um dos entraves dessa mudança, é o fato de que muitos programadores utilizam bancos de dados relacionais para persistir os dados, obrigando o programador a dominar linguagens SQL para realizar acessos ao banco de dados e realizar a persistência dos dados ou consultas no banco.

O mapeamento Objeto-Relacional utiliza-se de uma técnica bastante intuitiva no sentido que uma classe do tipo persistente pode ser mapeada para uma tabela do banco de dados relacional e atributos da classe para os campos da tabela (Figura1) [Chiarello 2008].

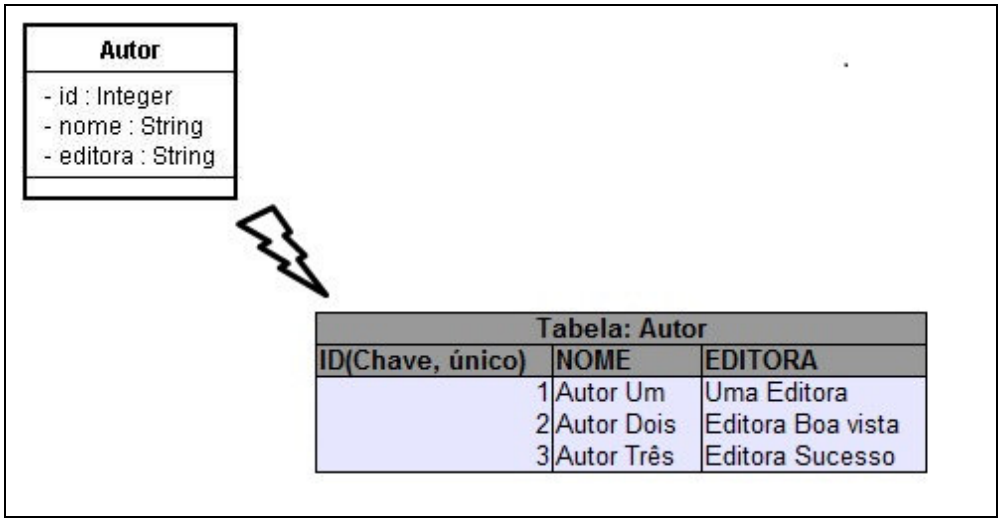

Figura 1. Exemplo de mapeamento Objeto-Relacional

2.1. Apresentação do Framework

Com o objetivo de auxiliar o desenvolvedor na implementação da camada de persistências, pretende-se apresentar uma opção que acredita ser viável e de fácil implementação em qualquer projeto baseado na tecnologia Java.

Com técnicas cada vez mais avançadas no mundo da programação Orientada a Objetos (OO), a persistência dos dados deve evoluir de forma semelhante. Existem opções que visam alcançar este objetivo, como o Hibernate, porém esta e outras opções possuem alguns inconvenientes como: difícil configuração e geralmente em inglês. O projeto Persiste é um framework que propõe uma forma simplificada de programação, permitindo ao desenvolvedor Java a possibilidade de utilizar um framework totalmente em português, com uma fácil configuração e integração ao sistema em desenvolvimento. O framework possui como base de funcionamento duas APIs Java, a Java Annotation e Java Reflection, permitindo ao framework realizar a introspecção dos objetos mapeados na aplicação através das anotações que serão descritas em breve.

A integração ao sistema ocorre de forma natural, sem a necessidade de utilização de arquivos XML, utilizando ao invés disso anotações para definição de tabelas, campos, chaves primarias e relacionamentos entre as classes, bem como outras opções importantes para fins didáticos e de depuração, como apresentar os comandos SQL executados, e as mensagens de depuração na saída padrão do aplicativo Java.

A única configuração que dispensa o uso de anotações é a conexão ao banco de dados, que é definida diretamente sobre a classe “persiste.banco.Banco”, a qual recebe informações como qual o Sistema Gerenciador de Banco de Dados utilizado, suportando de forma transparente Firebird e PostgreSQL, nesta classe também são informados dados importantes como o usuário, senha e endereço de conexão ao banco de dados.

Para o correto mapeamento objeto-relacional, a utilização das anotações torna-se necessária, as anotações possuem parâmetros que definem ou modificam a forma de funcionamento do framework.

A seguir será apresentada uma tabela com as anotações presentes no framework, juntamente com uma breve descrição sobre suas características.

Tabela 1: Anotações disponíveis no Framework

@AnotacaoTabela	O requisito básico para o funcionamento do framework é o relacionamento da classe com a tabela do banco de dados. A anotação AnotacaoTabela deve ser a primeira anotação a ser definida em cada classe, é ela a responsável pela identificação ou o mapeamento de uma tabela dentro da aplicação.
@AnotacaoChavePrimaria	Quando o framework deve persistir uma tabela que possui como chave primária um ou mais campos diferentes de ID, deve ser declarada a anotação AnotacaoChavePrimaria.
@AnotacaoColuna	Quando é utilizada uma variável de nome diferente da encontrada no banco de dados e o framework deve mapear a variável “x” com o campo “y” pode ser utilizado a AnotacaoColuna.
@AnotacaoChaveEstrangeira	Quando uma classe possui uma associação, e essa associação existe como uma chave estrangeira no banco de dados, deve ser utilizado a anotacaoChaveEstrangeira para realizar este mapeamento.
@AnotacaoChaveEstrangeiraCampos	Apesar do framework saber que uma classe qualquer possui um relacionamento com outra classe, ainda deve ser informado qual ou quais variáveis associam um objeto ao outro, esta associação é definida ao utilizar a anotação AnotacaoChaveEstrangeiraCampos.
@AnotacaoMapearCamposBD	Com a finalidade de facilitar o processo de desenvolvimento o framework mapeia automaticamente as variáveis com o mesmo nome que os campos da tabela no banco de dados, caso o programador não possa utilizar deste recurso ele deve utilizar a anotação AnotacaoMapearCamposBD.
@AnotacaoMostrarSQL	Esta anotação não é obrigatória, e não possui parâmetros, porém quando é utilizada, todas as operações efetuadas no banco de dados, terão os seus comandos mostrados na saída padrão do aplicativo.
@AnotacaoDepuracao	Esta anotação não é obrigatória, e possui como parâmetro um numero inteiro “nivel”, que definirá o nível de detalhe que o framework mostrará na saída padrão do aplicativo, este padrão é de 0 a 3, onde 0 não apresentará saída, e o 3 é o nível mais detalhado.

2.2 Exemplo de Utilização do Framework

Para o melhor entendimento, será apresentado um exemplo utilizando duas entidade, "Autor" e "Editora". Com a modelagem do banco de dados, e o diagrama de Classes.

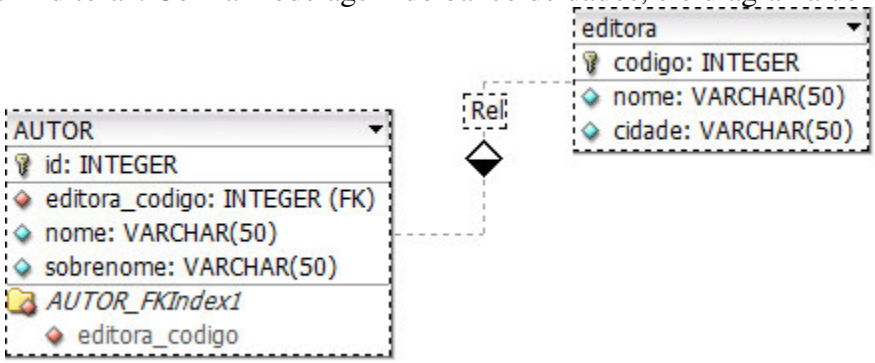

Figura 2. Modelagem do bando de dados de exemplo

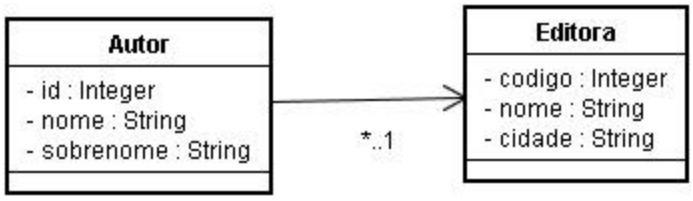

Figura 3. Diagrama de Classes

2.3 Implementação das Classes

```
1. public class Autor {
2.       @AnotacaoTabela(nome="autor")
3.       @AnotacaoMostrarSQL
4.       @AnotacaoDepuracao(nivel=5)
5.       Integer id;
6.       String nome;
7.       String sobrenome;
8.       @AnotacaoChaveEstrangeiraCampos(
9.                                  nome="editora_codigo",
10.                                 chaveRelacao="codigo",
11.                                 objetoRelacao="editora"
12.                                       )
13.      Integer editoraCodigo;
14.      @AnotacaoChaveEstrangeira(
15.                            carregarImediatamente=true,
16.                            salvarImediatamente=true
17.                                    )
18.      Editora editora;
19.      /*gets e sets */
20.      }
```

```
1. public class Editora {
2.      @AnotacaoTabela(nome="editora")
3.      @AnotacaoMostrarSQL
4.      @AnotacaoDepuracao(nivel=5)
5.      @AnotacaoMapearCamposBD(fazerMapeamento=false)
6.      @AnotacaoChavePrimaria(
7.                  nome="codigo",
8.                  usaNaInsercao=false,
9.                  usaNaAtualizacao=true,
10.                 usaNaPesquisa=true)
11.     Integer codigo;
12.     String nome;
13.     @AnotacaoColuna(nome="cidade")
14.     String CidadeEditora;
15.     /*gets e sets*/
16.     }
```

2.3 Manipulando Objetos

Este capítulo exemplificará sobre a utilização em tempo de execução do framework, com os seus principais métodos de persistência de objetos.

Após a carga dos dados no objeto a ser persistindo, no momento em que a informação deve ser salva, invoca-se o método persiste, pertencente à classe Crud. Este método possui como parâmetro o próprio objeto a ser persistido.

```
1. new Crud(objetoAutor.getClass()).persiste(objetoAutor);
```

Quadro 1: Persistindo objetos no banco de dados

Para carregarmos um objeto do banco de dados utilizamos os métodos carregar ou listar onde o método listar retorna um novo objeto, enquanto o método carregar preenche as informações do objeto em uso, conforme visto no Quadro 2

```
1. Autor objetoAutor = new Autor(5);
2. new Crud(objetoAutor.getClass()).carregar(objetoAutor);
3. //Ou ainda
4. Autor objetoAutor = new Autor(5);
5. Autor novoObjetoAutorMesmaID = crud.listar(objetoAutor);
```

Quadro 2: Carregando Objetos do Banco de Dados

Para excluir um objeto, é necessário que todos as variáveis referentes ao campo chave da tabela estejam preenchidos. Deve-se chamar o método exclui, que possui como parâmetro o objeto a ser excluído da base de dados.

```
1. new Crud(objetoAutor.getClass()).exclui(objetoAutor);
```

Quadro 3: Excluindo objetos do banco de dados

3. Avaliação do Framework

Para avaliar o framework, foi apresentado o mesmo à uma turma do curso de Sistemas de informação cursando a 7º fase, onde foi feita a apresentação sobre todos os aspectos do framework, bem como todos os seus métodos e anotações. Após a apresentação, foi desenvolvido um aplicativo de exemplo com duas classes "autor" e "editora" bem como os métodos de criação, exclusão, persistência e listagem de dados do banco de dados.

Após a conclusão do aplicativo utilizando o framework Persiste, foi aplicado um questionário de avaliação para os acadêmicos, baseado em uma visão geral da avaliação de usabilidade. O questionário utilizado foi adaptado para a realização da avaliação proposta onde as perguntas foram baseadas em uma escala de Likert, no que consiste em questões que utilizam uma escala de concordância, onde uma afirmação é feita e o avaliador assinala o grau de concordância com tal afirmação (Discordo completamente, Discordo, Neutro, Concordo, Concordo completamente) [Broke, 1986].

A avaliação foi respondida por 21 acadêmicos, sendo que cada questionário respondido continha 12 afirmativas. Em apenas 1 (um) questionário na afirmativa "Eu penso que precisaria de ajuda para desenvolver um sistema utilizando este framework.", um dos entrevistados não respondeu qual o nível de concordância à afirmativa em questão.

Com relação aos resultados obtidos na avaliação demonstrou-se que 76,19% dos entrevistados concordam em utilizar o framework frequentemente, e 14,29% concordam plenamente em utilizar o framework frequentemente. Na afirmação "Achei este framework extremamente simples", 42,86% concordaram, e 38,10% concordaram completamente com a afirmativa.

A Tabela 2 expõe os resultados da avaliação entregue aos acadêmicos em forma de porcentagem.

	Discordo completamente	Discordo	Neutro	Concordo	Concordo Completamente
1. Eu penso que gostaria de usar este *framework* freqüentemente.	0,00%	0,00%	9,52%	76,19%	14,29%
2. Achei este *framework* extremamente simples.	0,00%	0,00%	19,05%	42,86%	38,10%
3. Achei que foi fácil usar este *framework*.	0,00%	0,00%	0,00%	66,67%	33,33%
4. Eu penso que precisaria de ajuda para desenvolver um sistema utilizando este *framework*.	0,00%	35,00%	50,00%	15,00%	0,00%
5. Achei fácil definir qual anotação utilizar em cada local.	4,76%	0,00%	28,57%	42,86%	23,81%

6. Achei difícil configurar a conexão ao banco de dados.	38,10%	47,62%	9,52%	0,00%	4,76%
7. Eu imagino que a maioria das pessoas aprenderia a usar este *framework* rapidamente.	0,00%	0,00%	4,76%	61,90%	33,33%
8. Achei as anotações muito incômodas de usar.	42,86%	47,62%	9,52%	0,00%	0,00%
9. Eu me senti muito seguro(a) utilizando este *framework*.	0,00%	4,76%	38,10%	47,62%	9,52%
10. Eu precisei aprender muitas coisas antes de utilizar este *framework*.	14,29%	47,62%	28,57%	9,52%	0,00%
11. Este *framework* me ajudaria muito ao desenvolver um sistema.	0,00%	0,00%	4,76%	66,67%	28,57%
12. Este *framework* ajuda as pessoas que iniciam na área de programação.	14,29%	0,00%	0,00%	76,19%	9,52%

Tabela 2: Resultado da avaliação do framework

4. Conclusão

Para o desenvolvedor, o mais importante em uma aplicação é a regra de negócios de sua aplicação, porém boa parte de seu tempo é gasto realizando a persistência dos dados. Com a utilização do framework Persiste a persistência dos dados foi automatizada, resultando em um maior rendimento por parte do desenvolvedor na escrita do código do sistema. Toda a tarefa de inserção, atualização, leitura e exclusão dos dados é realizada por uma classe conhecida pelo nome “Crud”. Para isto o programador deve instanciar a classe Crud e invocar os métodos “persiste”, “exclui”, “listar”, passando como parâmetro a classe ou objeto que representa a tabela no banco de dados e no método listar incluir os filtros de dados.

O framework foi avaliado através da aplicação de um questionário que avaliava a usabilidade, baseado na escala de Likert, onde através deste, ficou claro que a maioria dos entrevistados considerou utilizar o framework novamente.

Como aprimoramentos ainda deste trabalho foram analisadas sugestões feitas em conjunto com colegas e professores, e desenvolvedores sobre as lacunas não preenchidas pelo framework, entre elas destaca-se a necessidade de implementar o suporte a associações N-N e 1-N, bem como de melhorar o suporte a associações 1-1. Outra característica necessária, é que o framework seja capaz de identificar quando não ouve alterações no banco de dados quando foi requerida a alteração, exclusão ou ainda inserção de dados, atualmente o controle é feito apenas baseado no retorno de erros por parte do banco de dados.

7. References

BROOKE, John. (1986) “Digital Equipment Corporation. SUS - A quick and dirty usability scale”. United Kingdom. Disponível em: <http://www.dis. uniroma1.it/~bertini/ ipc04/materiale/quest/sus.pdf>. Acessado em: 17 jun 2010.

CHIARELLO, Marcos. (2009) “Guia para Mapeamento Objeto Relacional Metodologia Celepar”. Disponível em: <http://www.documentador.celepar.pr.gov.br/ documentador/acessoPublico.do?action=downloadArquivoUuid&uuid=@gtf-escriba@624abe66-4662-47fa-a33e-1f90de2436df>. Acesso em: 30 out. 2009.

DAVID, Marcio Frayze. (2007) “Programação Orientada a Objetos: uma introdução”. Disponível em: <http://www.guiadohardware.net/artigos/programacao-orientada-objetos/>. Acesso em: 11 dez. 2009.

LEWIS, Daniel. (2008) Construindo Operações CRUD da Semantic Web Usando PHP. Disponível em: <http://www.ibm.com/developerworks/br/library/os-php-crud/ index.html>. Acesso em: 25 jul. 2009.

MACORATTI, José Carlos. (2003) “Padrões de Projeto : O modelo MVC - Model View Controller”. Disponível em:<http://www.macoratti.net/vbn_mvc.htm>. Acesso em: 25 jul. 2009.

PRADO, Mateus. (2009) “Persistência Relacional para Java e .NET com Hibernate”. Disponível em <http://mateusprado.com/blog/?p=21>. Acesso em: 24 jul. 2009.

REGO JUNIOR, Nilson de Souza. (2006) “Persistência de Dados em Hibernate”. Disponível em: <http://www.ime.usp.br/~kon/MAC5715/slides/Persistencia Hibernate.ppt>. Acesso em: 24 jul. 2009.

ROSA, Luíz Henrique de Araújo. (2003) “Framework em PHP – Projeto Rooda Devel”. Disponível em: <http://www.inf.unisinos.br/~barbosa/paradigmas/consipa3/61/artigos/ a14.pdf>. Acesso em: 2 ago. 2009.