FRAMEWORK PARA DESENVOLVIMENTO RÁPIDO DE SISTEMAS DISTRIBUÍDOS COM TECNOLOGIA .NET
XII INIC / VIII EPG - UNIVAP 2008

Alaor Bianco Rodrigues

Universidade do Vale do Paraíba – Faculdade de Ciência da Computação
Av. Shishima Hifumi, 2911 CEP 12244-000 São José dos Campos – SP – Brasil,
alaorbianco@yahoo.com.br

Resumo - O presente projeto destina-se à criação de um framework para desenvolvimento rápido e padronizado de aplicações específicas com acesso a banco de dados. O framework em questão facilita o desenvolvimento abstraindo as funcionalidades semelhantes entre sistemas e transparentemente implementando uma arquitetura de três camadas fazendo uso dos conceitos de orientação a objetos oferecidos pela tecnologia Microsoft .NET. O projeto possui uma alta interação e integração com SGBD, permitindo um grande dinamismo para implementar regras e consistências e ao mesmo tempo oferecendo uma alta flexibilidade para implementar uma série de soluções.

Palavras-chave: Framework, .NET, Sistemas Distribuídos, Orientação a Objetos
Área do Conhecimento: Ciências Exatas e da Terra - Ciência da Computação

Introdução

O software sob a ótica da Engenharia de Software é um produto e, portanto, espera-se dele características e comportamento que o leve a um determinado grau de qualidade. Características, estas, citadas abaixo:

- Funcionalidade
- Manutenibilidade
- Usabilidade
- Eficiência
- Confiabilidade
- Portabilidade

A adoção de uma tecnologia de desenvolvimento atualizada, bem como uma filosofia de desenvolvimento de um software também atualizada, proporciona qualidade e produtividade muito superiores para as empresas.

Em uma empresa, quando se há a necessidade de desenvolver sistemas específicos para atender suas necessidades de forma rápida garantindo uma padronização destes sistemas, e ainda, que se cumpram requisitos de qualidade é viável o desenvolvimento de um framework.

Metodologia

Um framework é um conjunto de classes com funcionalidades e comportamentos bem definidos que visam uma solução comum para problemas semelhantes.

Se analisarmos um software, ele possui um conjunto de características comuns, como acesso a dados, execução de procedimentos, apresentação dos dados. E até mesmo as telas do sistema possuem características comuns nos distintos estados que podem assumir, como inserindo, alterando, excluindo ou navegando.

Um framework busca empacotar todas essas características comuns em uma aplicação e abstraí-las deixando para ser implementado apenas as características específicas de cada sistema ou tela.

Arquitetura de Sistemas em Camadas

O desenvolvimento de sistemas em camadas é o cerne de aplicações distribuídas e agora, com o advento dos *Web Services*, o desenvolvimento em camadas rompe fronteiras ampliando as possibilidades de uso de componentes distribuídos separados pela Internet.

A arquitetura de sistemas adotada para esta solução é a de Três Camadas (*Three-Tier*):

1. Camada de Apresentação (*Presentation Tier*): interface gráfica com o usuário (*Web* ou *Desktop*) e lógica de apresentação
2. Camada Intermediária (*Middle Tier*): camada de acesso a dados, regras de negócio, e componentes compartilhados como autenticação e validação
3. Camada de Dados (*Data Tier*): Banco de Dados Relacional (Oracle)

Esta arquitetura apresenta algumas vantagens como:

- Uma dada informação ou serviço pode ser disponibilizado, tanto em aplicações Desktop quanto para Web, fazendo o uso das mesmas camadas de negócio e de dados, ou seja, alterando apenas a camada de apresentação.

- A aplicação cliente é liberada do processamento das regras de negócio e acesso a dados, podendo, a máquina do usuário, ter uma configuração modesta.
- Possibilidade de desenvolvimento e manutenção, de forma isolada, da interface com o usuário, das regras de negócio e do acesso a dados.

Programação orientada a objetos

A Orientação a Objetos, ou OO, é uma tecnologia para a produção de modelos que especifiquem o domínio do problema de um sistema.

Modelos orientados a objetos são implementados convenientemente utilizando uma linguagem de programação orientada a objetos. A engenharia de software orientada a objetos é muito mais que utilizar mecanismos de sua linguagem de programação, é saber utilizar da melhor forma possível todas as técnicas da modelagem orientada a objetos.

A orientação a objetos não é só teoria, mas uma tecnologia de eficiência e qualidade comprovadas, usada em inúmeros projetos e para construção de diferentes tipos de sistemas.

A orientação a objetos requer um método que integre o processo de desenvolvimento e a linguagem de modelagem com a construção de técnicas e ferramentas adequadas.

A técnica de desenvolvimento orientado a objetos proporciona as seguintes vantagens:

- Por requerer divisão de responsabilidades na modelagem dos objetos, torna o sistema bastante flexível a mudanças. Na maioria das alterações que envolvem alteração em todo o sistema, basta alterar apenas algumas classes.
- Proporciona a criação de objetos e componentes genéricos e totalmente reutilizáveis. Através da OO, é possível criar outros objetos herdando propriedades e métodos de outros componentes. Isto proporciona padronização de código-fonte e interface gráfica.
- A reutilização, proporcionada pela OO, possibilita grande aumento de produtividade, pois com ela é possível amortizar o custo do projeto utilizando componentes e classes desenvolvidas em projetos anteriores ou mesmo no projeto atual.
- Além disso, a reutilização aumenta o nível de qualidade do desenvolvimento, pois ao utilizar um componente ou classe já utilizada e testada com sucesso em outros sistemas desenvolvidos, segundo a OO, diminui-se o índice de bugs e falhas.

Tecnologias

A tecnologia .Net nos proporciona as ferramentas necessárias para desenvolver esse framework. Foi escolhifo o .Net Framework 2.0 para fins de compatibilidade com versões antigas do sistema operacional Windows anteriores ao XP.

A próxima escolha foi quanto à tecnologia de acesso a dados e de distribuição desse acesso.

Foi adotado a tecnologia .Net Remoting por não haver problemas de interoperabilidade uma vez que todo o sistema usará a tecnologia .Net e por ser mais eficiente nos quesitos velocidade e integridade uma vez que pode trafegar dados em formato binário sobre o protocolo TCP.

Como Sistema Gerenciador de Banco de Dados (SGBD) utilizou-se o Oracle 10g.

Acesso a Dados

O SGBD foi instalado numa máquina servidora dedicada (servidor de banco de dados), e em uma outra máquina (servidor de aplicação) foi instalado um cliente de banco de dados Oracle e desenvolvido um Serviço Windows (*Windows Service*) que irá receber todas as requisições de acesso a dados e processá-las.

Este serviço windows registra um tipo conhecido (OracleBD) que é a classe proxy que o cliente irá usar e cria uma porta TCP para receber as requisições usando uma *Thread* para cada uma que receber.

```
RemotingConfiguration.RegisterWellKnownService
Type(GetType(OracleBD),ccObjectURI,
WellKnownObjectMode.SingleCall)
ChannelServices.RegisterChannel(New
TcpServerChannel(ccPortaIP), False)
```

Para manter as classes desconectadas o cliente não acessa diretamente a classe OracleBD e sim uma interface que esta classe implementa.

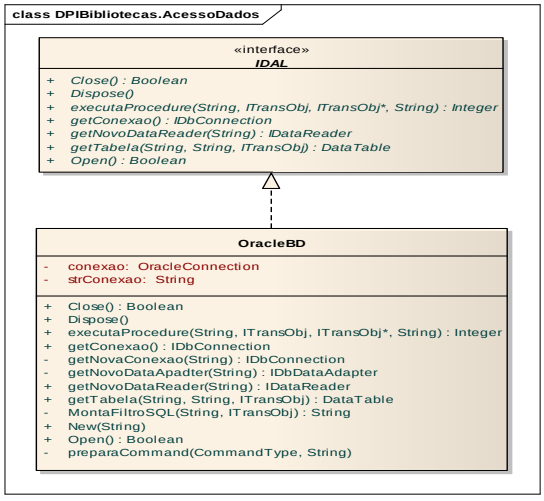

Figura 1: Classe de Acesso a Dados

Do lado cliente, que irá consumir o serviço, o objeto remoto é criado da seguinte forma:

```
Dim oBD as IDAL
oBD = DirectCast(Activator.GetObject(GetType(
OracleBD),ccEnderecoServico),IDAL)
```

Onde IDAL é a interface implementada pela classe de acesso a dados. Isso cria uma objeto proxy (remoto) da classe OracleBD.

Componentes

Foram desenvolvidos componentes específicos para usar com este framework de modo a facilitar a manipulação dos mesmos e facilitar também o vínculo com uma fonte de dados. Todos os componentes que exibem dados implementam a Interface IDPIComponentes para generalizar o tratamento com eles.

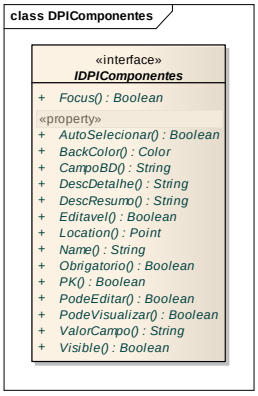

Figura 2: Interface para os controles de tela

Todos os componentes que implementam essa interface possuem uma propriedade CampoBD que informa a qual campo no banco de dados ele está relacionado.

Há um componente container chamado DPIPainel, que agrupa os controles relacionado a uma determinada fonte de dados e possui eventos antes e depois de cada operação de inserir, alterar excluir etc. Desta forma pode-se, por exemplo, validar se as condições necessárias para o evento excluir estão satisfeitas e liberar ou não a sua execução e após a execução realizar todas as operações necessárias e assim também como cada outro evento.

Outro componente essencial para o projeto é a estrutura de dados do tipo dicionário que armazena uma referencia para cada controle dentro do container chamada Controles.

Esta estrutura permite uma recuperação rápida do objeto pelo nome ou pelo campo que se relaciona com o banco de dados (propriedade CampoBD).

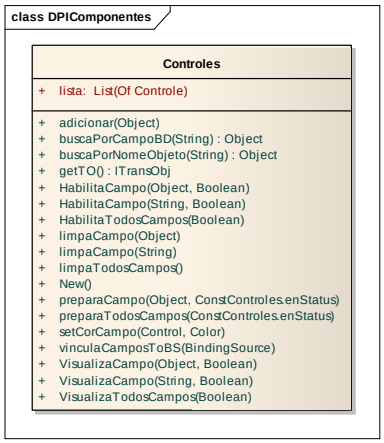

Figura 3: Classe Controles

Apresentação

A camada de apresentação é Desktop (*Windows Forms*) e se preocupa apenas com a interface gráfica com o usuário, ou seja, apenas com a lógica de exibição dos dados uma vez que as consistências serão realizadas na camada de negócio.

Na carga das telas é realizada uma consulta ao dicionário de dados do SGBD que irá trazer as informações dos campos das tabelas envolvidas nesta tela como tamanho máximo, obrigatoriedade, se é uma chave primária, entre outras. Essas propriedades são configuradas nos controles e as regras de preenchimento são validadas, por exemplo, se todos os campos obrigatórios estão preenchidos, se um determinado campo possuir um *check constraint* seu valor será validado.

No modo de inserção ou alteração os campos obrigatórios são automaticamente destacados com uma cor diferente de fundo.

Regras de Negócio

Todas as regras são implementadas nesta camada e as validações independem do nome do objeto na tela, são todas realizadas baseadas no nome do campo no banco de dados.

Há uma forma automatizada de validar as regras de negócio, há uma tabela no banco de dados chamada regra_consistir que possui a seguinte estrutura:

Campo	Operador	Valor	Contexto	Seq
Login	<>	NULL	Usuário	1
Valor	>	0	Venda	2

A interpretação é a seguinte: no contexto usuário há uma regra de que o campo Login deve ser diferente de nulo, ou no contexto Venda há uma regra que diz que o campo valor deve ser maior do que zero.

É informado qual contexto se quer consistir e recupera-se do banco de dados as regras a serem consistidas. Através do nome do campo se recupera o controle relacionado na tela e, por conseguinte, o valor nele contido e verifica-se a condição. Caso a condição não seja satisfeita pode-se emitir um alerta e impedir a progressão da execução.

Comunicação entre Camada de Apresentação e Negócio

Toda a comunicação entre a camada de apresentação e negócios é feita por um objeto do tipo TransferObject (TO) que é uma estrutura de dados do tipo dicionário que armazena o nome do campo relacionado ao banco de dados e o valor de suas propriedades.

A camada de negócio realiza as consistências e se houver erros ela devolve um código de erro pré-determinado para a camada de apresentação que será interpretado e realizado as devidas medidas.

Baseado no TO recebido, a camada de acesso a dados pode executar uma procedure, executar um comando ou qualquer outra operação.

Foi convencionado que os parâmetros da procedure no SGBD devem possuir o nome do campo relacionado precedido da letra p, por exemplo, o campo usua_login será preenchido pelo parâmetro pUsua_Login.

A camada de dados deriva os parâmetros de um command e popula sua coleção de parâmetros.

OracleCommandBuilder.DeriveParameters(cm)

Os valores dos parâmetros são automaticamente preenchidos baseado no TO passado para a camada de acesso a dados. Caso haja parâmetros de retorno da procedure um novo objeto TO é construído e devolvido para quem o chamou.

Resultados e Discussão

O primeiro resultado percebido foi a velocidade com que se desenvolve uma nova tela ou mesmo uma nova aplicação.

Outro ganho significativo foi o fato de mais de uma pessoa poder trabalhar simultaneamente numa mesma tela, uma com regras e outra com a apresentação dos dados. Para o ambiente operacional do sistema, onde alterações de regras de negócios é com freqüência uma alteração é minimizada uma vez que se, em suma maioria, faz necessário alterar apenas uma linha de registro no banco de dados sem a necessidade de recompilar e redistribuir a aplicação e seus componentes.

A curva de aprendizado se demonstrou muito mais comportada, tendendo à estabilidade muito mais cedo.

Arquiteturalmente separar as camadas lógicas facilitou muito a manutenção, e a consolidação da aplicação com o tempo e sua estabilidade garante a qualidade dos futuros projetos.

Conclusão

Com a implantação do framework e análise dos resultados obtidos o projeto é considerado de extremo sucesso. Todos os objetivos foram alcançados e a expectativa foi superada. Todos os requisitos funcionais foram implementados e todas as funções semelhantes foram empacotadas e abstraídas do desenvolvimento.

As boas práticas de desenvolvimento adotadas bem como o paradigma de orientação a objetos proporcionam grande ganhos e infinitas possibilidades de evolução.

A adoção de aplicações distribuídas abre um novo horizonte para o provimento de serviços em vários ambientes e varias plataformas bem como a fácil integração entre sistemas.

Referências Bibliográficas

RATIONAL, Publicações eletrônicas, The UML and Data Modeling, www.rational.com

PIALORSI, PAOLO E RUSSO, MARCO, Introducing Microsoft LINQ, 2007 Microsoft Press

KIMMEL, PAUL, Advanced C# Programming, Editora McGraw-Hill/Osborne

URMAN, SCOTT, Oracle 9i: Programação PL/SQL, Editora Campus