Propósito

Este documento é um guia sobre o modelo arquitetural do plataforma de desenvolvimento da empresa BgmRodotec, para a plataforma .NET. Os aspectos tecnólogicos citados neste documento têm como base a plataforma .NET versão 3.5/3.5 Service Pack 1 e 4.0(com expectativa de lançamento para este ano de 2009).

Este documento não representa de forma alguma uma norma irreversível em termos intelectuais. Toda possibilidade de evolução deve ser considerada em momento comercial, tecnológico e financeiramente adequado. No entanto, os aspectos citados em cada versão deste documento, devem ser respeitados integralmente de acordo com o mais adequados comportamentos ético e profissional, assim como de respeito ao esforço dos envolvidos direta e indiretamente.

Este documento traz uma descrição e explicação dos aspectos de arquitetura de implementação da plataforma BgmRodotec.NET. Este documento também pode conter trechos de propostas arquiteturais experimentais e assim serão adequadamente destacados como tal para evitar seu uso de maneira inadequada em ambientes internos ou externos.

O termo "arquitetura de implementação", como utilizado neste contexto, representa como vários aspectos são definidos para que sejam normas de construção de sistemas baseados na citada. Os aspectos envolvem, mas não estão limitados, a:

* Padrões de codificação baseados no Microsoft Design Guidelines (ver tabela de referências).
* Padrões de implementação baseados no Microsoft Architecture Guide
* Padrões de implementação baseados em tecnologias disponíveis na plataforma Microsoft .NET
* Aspectos específicos para determinadas linguagens de programação. Por exemplo, modelos de código que são considerados inadequados e não serão utilizados.
* Aspectos da plataforma .NET, para que as escolhas sejam compreendidas e a qualquer momento possam ser reavalidas para mudança.
* Aspectos de negócio que determinaram a estrutura e organização da implementação.

Autores

* Paula Carolina Damasio Guelere - BgmRodotec Company
* Roger Emerson Alves Villela - Open Mind Company

**Conceitos**

Workflow(Fluxo de execução/Processo de negócio)

O termo workflow está sendo usado por mim, Roger Villela, de maneira provisória para representar o conceito de processo de negócio dentro de um contexto específico. Também utilizo alguns termos em inglês apenas temporáriamente. Neste trecho eu tento definir conceitos que podem facilitar a construção de um sistema mais flexível baseado na plataforma BgmRodotec.NET.

|  |  |
| --- | --- |
| Conceito | Descrição |
| Task | Uma **Task**(tarefa) representa uma unidade indivisível do ponto de vista do negócio(contexto). Se uma determinada tarefa é composta por outras tarefas, ela somente pode ser considerada completa, se todas as suas etapas foram realizadas. Se alguma etapa não pode ser realizada, ela deve ser considerada não realizada. Uma tarefa é aplicável ao contexto e não ao tipo, como uma classe que representa uma entidade. |
| Query Action | Uma **Query Action**(consulta), representa uma atividade que têm como função específica recuperar valores de uma fonte de informação. A **Query Action** não representa o mesmo conceito que uma pesquisa(**Find Action**). |
| Find Action | A **Find Action** (pesquisa) pressupõe uma atividade que pode ou não retornar algum resultado dependendo do estado das informações que foram oferecidas como critério de pesquisa e do estado das informações na origem. |
| Change Action | A **Change Action**(alteração) representa uma atividade que têm como propósito a realização de mudanças no estado das informações em sua origem. Ou seja, é uma mudança que será reconhecida por todas as partes do sistema que estão direta ou indiretamente relacionadas com a mesma. |
| Validation Rule Action | Uma **Validation Rule Action**(regra) representa uma atividade que aplica um ou mais critérios que têm por único e exclusivo propósito retornar um "valor" indicando se o critério foi aplicado com sucesso ou não. Por aplicado com sucesso ou não, considera-se o contexto. Existe a possibilidade de um grupo especializado desta atividade que pode realizar uma ação associada em objetos alvo. |
| System Action | Uma **System Action**(atividade de sistema) representa uma atividade que têm como princípio a ortogonalidade, ou seja, ela pode estar presente em qualquer parte do sistema, não afetando de maneira direta ou indireta as regras de negócio da aplicação que utiliza o framework. |
| Data Activity | Uma **Data** **Activity**(atividade) representa uma ação direta na fonte de dados, como um arquivo XML ou um banco de dados, por exemplo. Uma **Activity** apresenta um conceito que pode ser utilizado por uma **Action** e/ou por uma **Task**. O uso de uma **Data** **Activity**  diretamente em uma **Task** é algo que ainda não considero adequado e preciso conversar com a Paula sobre este uso. |
| BizEntity | Uma **BizEntity** representa um conceito de negócio de maneira individual. Uma **BizEntity** pode ter em sua composição outras entidades do tipo **BizEntity**. A **BizEntity**, não pode ter comportamento de negócio implementado. Ela pode ter, no entanto, comportamento necessário ao modelo de objetos da plataforma .NET, como a implementação do método ToString. Qualquer manipulação que possa interferir na ortogonalidade da **BizEntity** deve ser aplicado utilizando-se algum padrão de implementação e não diretamente na **BizEntity**. |
| Workflow | Um workflow é representado pelo conjunto de itens descritos acima. A forma como estes conceitos são utilizados dentro do workflow deve ser o mais flexível possível, presevando os conceitos e sua aplicação. |

Tabela 1.0 - Descrição resumida dos conceitos.

# Descrição do modelo

Este trecho do documento descreve o modelo dos conceitos citados acima.

**Figura - Modelo Conceitual**

**De acordo com este modelo a lógica de um processo de negócio funciona da seguinte forma.**

* **Todo processo de negócio têm uma tarefa.**
  + **Cada tarefa representa uma unidade indivisível para completar um requerimento do processo de negócio.**
* **A tarefa pode se utilizar de outras tarefas, para atingir seus objetivos.**
* **Cada tarefa, individualmente, é composta por uma ou mais ações de determinados tipos. O tipo da ação utilizada pela tarefa está classficado na Tabela 1.0. Dentro de uma tarefa, as ações podem ser combinadas livremente com outras ações para atingir o objetivo da tarefa.**
* **Cada ação pode se utilizar de uma atividade de tratamento de dados, que representa o acesso a fonte de informação, seja para recuperação ou alteração das informações. Uma ação pode se utilizar de uma ou mais atividades de tratamento de dados. A ação em momento algum reconhece qual o tipo de fonte de informação em sua estrutura original, por exemplo, XML. A atividade de tratamento de dados é responsável integral pela transferência da informação de seu formato original para a BizEntity que a ação necessita e também responsável pela transferência da BizEntity para o formato original da informação.**

**NOTA: Este processo de transferência de informação entre a atividade de tratamento de dados e o mecanismo de persistência ainda precisa ser trabalhado conceitualmente, através de alguns Design Patterns. Neste momento apenas a linha de raciocínio foi traçada, mas o modelo de implementação entre a atividade e as classes que realizam o tratamento das informações, como API XML ou API de banco de dados, ainda precisa ser definido e testado nos cenários que representam cada um dos 5 tipos de ação: Query, Find, System, Validation, Change.**

**Task - Conceito e aplicação**

Uma Task é representada através de uma entidade abstrata, seja uma classe ou um interface. No exemplo abaixo o conceito de Task têm dois comportamentos definidos em uma interface chamada ITask:

|  |  |
| --- | --- |
| Namespace | BgmRodotec.Globus.Comum |
| public interface ITask {  TType Execute<TType>();  void ExecuteWithoutReturn<TType>();  } | |

**Tabela 1.1 - Definição da interface ITask**

Uma Task, conforme conceitualizada brevemente na tabela 1.0, representa uma unidade indivisível. No entanto, o número de ações que estão contidas na atividade é indefinido e deve ser assim, pois cada contexto de uso exige um grupo de ações diferente. A classe que implementa a interface ITask deve conter uma lista de ações adequadas ao seu propósito. Por exemplo, se em um determinado workflow for necessário apenas ações do tipo Change. A classe pode então implementar livremente qualquer número de ações do tipo Change que sejam requeridas pelo processo de negócio e alterá-las também com tal flexibilidade. A interface ITask têm dois comportamentos:

* Execute: que executa um número arbitrário de ações que retorna algum tipo decorrente desta atividade, por exemplo, uma lista de itens.
* ExecuteWithoutReturn: que executa um número arbitrário de ações mas não retorna valor algum decorrente desta atividade.

Ambas têm em sua definição o uso de Generics para oferecer a flexibilidade requerida pelo modelo.

Para ilustrar a implementação de uma Task, segue abaixo o trecho de código de uma classe chamada **ListasPaises**. O nome da classe em si, não é relevante neste momento e também não reflete o padrão de nomes que é descrito mais adiante neste documento para uso adequado do idioma português neste contexto:

|  |  |
| --- | --- |
| Namespace | BgmRodotec.Globus.Workflow.Listas.Tasks |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace BgmRodotec.Globus.Workflow.Listas.Tasks {  using Comum;  using Action.Query;  using DataAccess;    public class ListarPaises: ITask {  List<IEntityQuery<Pais>> actions = new List<IEntityQuery<Pais>>();  #region ITask Members  /// <summary>  /// Executa um número de ações  /// de acordo com a lógica requerida pela tarefa.  /// </summary>  /// <typeparam name="TType">Tipo alvo que a aplicação  /// aplica as ações.</typeparam>  /// <returns></returns>  public TType Execute<TType>() {  TType result = default(TType);    actions.Add( new Paises() );  foreach ( var item in actions ) {  item.List();  }    return result;  }  public void ExecuteWithoutReturn<TType>() {  throw new NotImplementedException();  }  #endregion  #region Inner class  private class Paises : IEntityQuery<Pais> {  #region IEntityQuery<Pais> Members  IEnumerable<Pais> IEntityQuery<Pais>.List() {  // Qualquer classe dentro do workflow(module)  // representa uma sequência de passos que devem  // ser executados, como checagem de seguranção,  // para que a tarefa possa ser executada.  return new ListasFactory().ListadePaises();  }  IEnumerable<Pais> IEntityQuery<Pais>.ListWithSingleCriteria( SingleParameter<Pais> criteria ) {  IEnumerable<Pais> result = null;  if ( criteria == null )  throw new NullReferenceException( "O delegate não pode ser null" );  return result;  }  IEnumerable<Pais> IEntityQuery<Pais>.ListWithAditionalCriteria( MultipleParameter<Pais> criteria ) {  IEnumerable<Pais> result = null;  if ( criteria == null )  throw new NullReferenceException( "O delegate não pode ser null" );  return result;  }  #endregion  }  #endregion  }  } | |

Neste exemplo temos uma coleção List<T> onde T é uma classe/struct que implementa a interface IEntityQuery<TEntity>, onde TEntity, é qualquer classe/struct que implementa a interface IBizEntity. BizEntity neste contexto representa apenas é tão somente classes/structs do contexto de negócio. O termo não é utilizado para descrever qualquer classe nativa da plataforma .NET e sim classes/structs criadas para representar o contexto de negócio em qualquer aplicativo que utilize a plataforma BgmRodotec.NET. Também temos um inner class chamada Paises. A classes Paises representa um Query Action, cujo modelo é definido através da interface IEntityQuery<TEntity>.

|  |  |
| --- | --- |
| Namespace | BgmRodotec.Globus.Workflow.Action.Query |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace BgmRodotec.Globus.Workflow.Action.Query {  using Comum;  /// <summary>  /// Realiza uma query recebendo um parâmetro.  /// </summary>  /// <typeparam name="TEntity"></typeparam>  /// <returns></returns>  public delegate IEnumerable<TBizEntity> SingleParameter<TBizEntity>( TBizEntity criteria )  where TBizEntity : IBizEntity;  /// <summary>  /// Realiza uma query recebendo múltiplos parâmetros.  /// </summary>  /// <typeparam name="TEntity"></typeparam>  /// <returns></returns>  public delegate IEnumerable<TBizEntity> MultipleParameter<TBizEntity>( params TBizEntity[] criteria )  where TBizEntity : IBizEntity;    /// <summary>  ///  /// </summary>  /// <typeparam name="TEntity"></typeparam>  public interface IEntityQuery<TBizEntity> : IAction<TBizEntity> where TBizEntity : IBizEntity {  IEnumerable<TBizEntity> List();  IEnumerable<TBizEntity> ListWithSingleCriteria( SingleParameter<TBizEntity> criteria );  IEnumerable<TBizEntity> ListWithAditionalCriteria( MultipleParameter<TBizEntity> criteria );  }  } | |