BRQ

Teste Trades/Classificação de Risco.

## SOBRE A ABORDAGEM PARA ATENDER AOS QUESITOS DO TESTE

### Resumo

Your design must take into account category rules can be added/removed/modified and will become highly complex in the near future.

Please write your answer in pseudo-code showing clearly what classes, interfaces, methods and design patterns you would create/use to solve this problem. Also, object oriented programming is appreciated.

As regras são armazenadas numa estrutura que contempla:

Nome do Conjunto de Regras

Setor do Cliente (Publico, Privado, [outros])

Range de Valore <Min e Max>

Risco atribuído para a faixa (LOW, MEDIUM, HIGH, [NONE – caso não haja regra aplicável, [outras classificações].

O algoritmo é:

Para Cada Trade

Selecionar o sub-conjunto de regras aplicável ao setor do Trade.

Ordenar as regras pelo Valor Minimo em ordem crescente

Atribuir NONE para a Classificação

Para Cada Regra

Testar o trade para todas as regras

Sendo a Regra Aplicavel Atribuir a Classificação

Prevalece a ultima classificação atribuída.

OBS: É interessante entender mais sobre o processo de classificação de risco para poder desenvolver um algoritmo que atenda realmente Às necessidades.

No algoritmo acima é facilmente parametrizável comportamentos como: sequencia de aplicação de regra, interrupção na primeira ou na ultima aplicação da regra.

Contudo havendo complexidade maior, pode ser necessário adoção de mecanismo de “interpretação” de regras ou “expressões”, ou mesmo compilação de “regras” através de ferramentas como o Rosylin.

Havendo a possibilidade de os parâmetros de entrada (Trades) terem mais variáveis, será necessário rever interfaces e outros elementos. Caso contrário outro método de classificação poderá ser facilmente injetado.

Para implementar o app, utilizou-se:

DDD,

IoC

ORM (EF),

MVC

Optou-se por persistir todos os dados em MD, inclusive os Trades. O que se entende ser desnecessário em um ambiente adequado (produção).

## Sobre as regras:

Entendendo que os parametros de entrada SEMPRE serão Valor e Setor, por conta da interface apresentada, temos que as regras possíveis podem ser generalizadas como:

ValorMinimo, ValorMaximo, Setor, RISCO onde RISCO é a Classificação, Que neste caso também poderá ser as previamente especificadas, e também novas classificações

Observação: Poderemos entender que minimo ou maximo nulos ou zeros significam sem limite. Podemos também adotar numeros absolutos para o máximo.

Teriamos então para os quesitos do teste as seguintes possibilidades:

0, 1.000.000, PUBLIC, LOWRISK

1.000.001, 999.999.999.999, PUBLIC, MEDIUMRISK

1.000.001, 999.999.999.999, PRIVATE, HIGHRISK

ATENÇÃO: Dentre estas regras definidas, parece faltar regra para setor PRIVADO até 1.000.000.

Estas regras podem ser mantidas em Banco de Dados. O Método de Classificação recebe o conjunto de regras como parâmetro.

a Estrutura da Classes contemplará:

Class GrupoDeRegras {

int GrupoDeRegrasId // chave primaria

string Descricao

}

Class Regras {

int RegraId // chave primaria double MinValue

double MaxValue

string ClientSector

string Risk

int GrupoDeRegrasId // Foreign Key

}  
  
A Interface de chamada da classificação é a determinada:

interface Trade {

double Value { get; }

string ClientSector { get; }

}

A tabela GrupoDeRegras serve par agrupar um conjunto de regras.

A classificação do Trade se dará aplicando sequencialmente as Regras, e a ultima regra valida aplicada será a retornada.

Devido a falta de definição de uma classificação para situações não contempladas, será retornado "NONE" como classificação para o Trade. *Entendo que é necessário uma definição para tal situação.*

A sequência de aplicação das regras será do menor ValorMinimo para o maior Valor Minimo, isso devidamente filtrado para o ClientSector da trade.

A abordagem permite que caso de vários critérios de classificação possam ser "testados" até se conseguir um "critério" adequado, bastara inserir novos GruposDeRegras, e para cada um destes as regras necessárias.

As regras permitem que o ClientSector também possa ser "Public", "Private" ou outro que venha a ser necessário.

A classificação retornada poderá ser "LOWRISK", "MEDIUMRIS", "HIGHRISK" ou qualquer outra que venha a ser necessária.

Como a especificação determina apenas Value e ClientSector para o Trade, apenas estes critérios foram utilizados, porém caso o critério seja acrescido de mais parâmetros, bastará alterar os Dominios.

A tabela Regras contempla cada regra a ser utilizada para classificar um trade.

A classificação do Trade se dará aplicando sequencialmente as Regras, e a ultima regra aplicada será a retornada.

A sequência de aplicação das regras será do menor ValorMinimo para o maior Valor Minimo, isso devidamente filtrado para o ClientSector da trade.

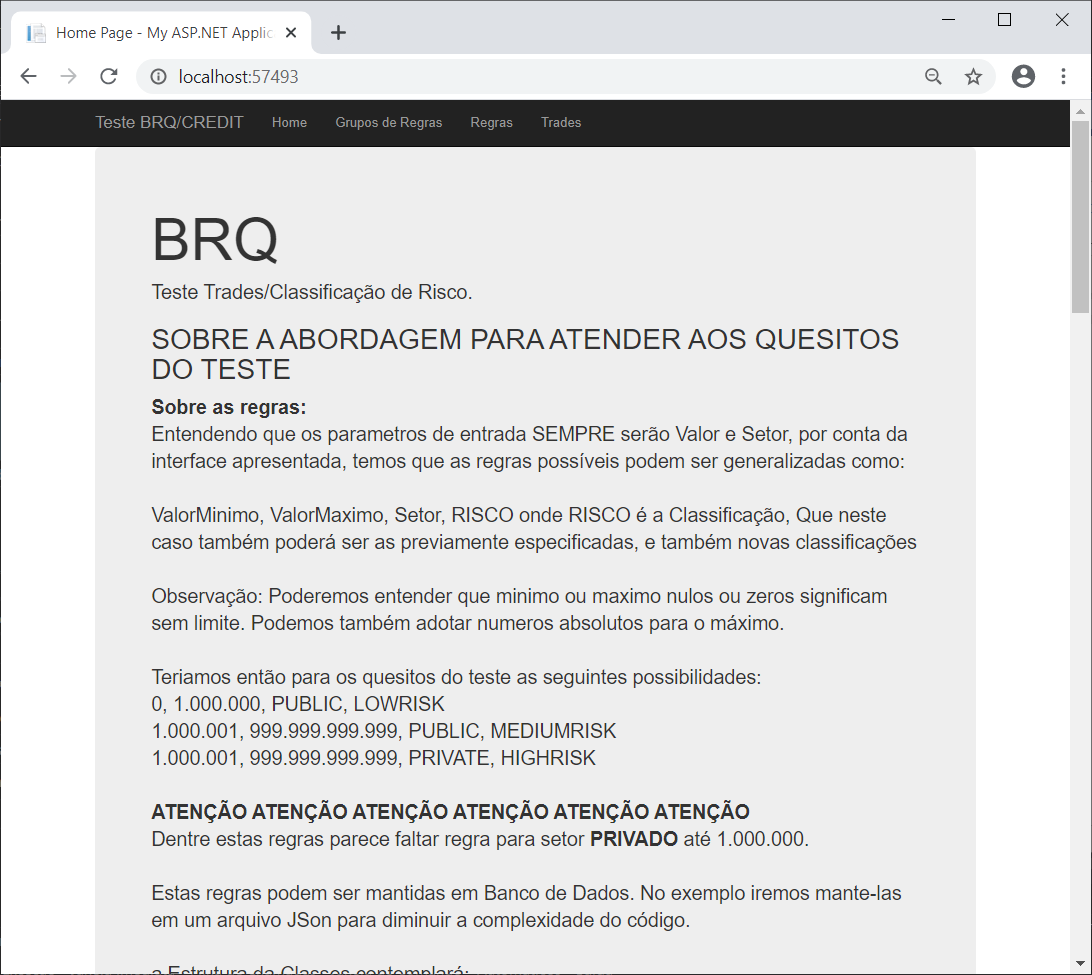
Como aprimoramento do mecanismo de classificação entendo que poderia ser interessante uma definição explicita da sequência de aplicação das regras. O que implicaria em acrescentar um atributo Sequencia na classe de Regras.

A abordagem permite que caso de vários critérios de classificação possam ser "testados" até se conseguir um "critério" adequado, basta´ra inserir novos GruposDeRegras, e para cada um destes as regras necessárias.

As regras permitem que o ClientSector também possa ser "Public", "Private" ou outro que venha a ser necessário.

A classificação retornada poderá ser "LOWRISK", "MEDIUMRIS", "HIGHRISK" ou qualquer outra que venha a ser necessária.

Como a especificação determina apenas Value e CleintSector para o Trade, apenas estes critérios foram utilizados, porém caso o critério seja acrescido de mais parâmetros, bastará alterar os Dominios.



# Teste BRQ/Credit

Projeto foi criado em ASP.NET MVC, utilizando .Net Framework, DDD, EF (apenas para poder persistir as regras em momento oportuno), IoC e AutoMapper

**# Execução**

Para compilar e rodar o app poderá ser necessário a instalação do ef, para tanto no Console do Gerenciador de Pacotes execute

PM> Install-Package EntityFramework

**# Descritivo**

Este é um projeto que tem foco no IoC, e que pretende demonstrar como "Regras de Classificação" com base em 2 valores podem ser utilizadas de forma dinâmica, utilizando algumas tecnologias e padrões de projetos:

 ASP.NET MVC para front end

 .Net Framework - para o core do app (poderia ser adotado também o net.core)

 DDD (Domain Drive Design) - Como pattern para desacoplamento)

 IoC (Simple Injector)

 Fluent API

 Linq/Lambda para filtragens

 EF para manter as regras - utilizando SQLServer.

**# Observação**

Para diminuir o tempo necessário foi utilizado um modelo de aplicação previamente preparado, como base da estrutura e alguns outros itens acessórios.

## Telas do App desenvolvido para o Teste.

### Manutenção dos Conjuntos de Regras

## 

### Classificação dos Trades Cadastrados

Utilizado para testes, chamando o método de Classificação, que utiliza o Conjunto de Regras da linha.

## 

### Manutenção das Regras

## 

### Cadastramento de Trades para Fins de Testes

## 