Laborator 4.1

1. Validation Application Block

Sursa: Microsoft Enterprise Library

Scopul blocului: pune la dispozitie un set de clase care permit validarea tipurilor de date utilizator. De exemplu, se poate verifica daca o variabila de tip referinta este nenula, daca un sir de caractere are o anumita forma (expresie regulata) sau are lungime minima/maxima sau daca un numar este continut intr-un anumit interval.

Specificarea validatorilor:

- prin fisier de configurare
- *prin atribute*
- prin cod

Se poate folosi pentru:

- Biblioteci (clase)
- ~~Windows forms~~, aplicatii web ASP.NET MVP, WPF
- Windows Communication Foundation

Exemplu

```
//fisier Customer.cs
using Microsoft.Practices.EnterpriseLibrary.Validation.Validators;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _1.DemoValidationCustomer
{
    public class Customer
    {
        private string _name;

        //public String Name
        //{
        //    get
        //    {
        //        return _name;
        //    }
        //    set
        //    {
        //        if (value == null || value.Length < 3 || value.Length > 50)
        //        {
        //            throw new Exception("The name is null or too short or too
long");
```

```
        //         }
        //         else
        //         {
        //             _name = value;
        //         }
        //     }
        //}

        [NotNullValidator(MessageTemplate="The name cannot be null")]
        [StringLengthValidator(3, RangeBoundaryType.Inclusive, 50,
RangeBoundaryType.Inclusive, MessageTemplate="The name should have between {3} and {5}
chars")]
        public String Name
        {
            get;
            set;
        }
    }
}

//fisier Program.cs
using Microsoft.Practices.EnterpriseLibrary.Validation;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _1.DemoValidationCustomer
{
    class Program
    {
        static void Main(string[] args)
        {
            Customer customer = new Customer();
            customer.Name = null;
            //customer.Name = "Aa";
            //in the naive implementation: Exception thrown
            ValidationResults validationResults = Validation.Validate<Customer>(customer);
            if (validationResults.Count == 0)
            {
                Console.WriteLine("The object is fine");
            }
            else
            {
                Console.WriteLine("There are some issues: ");
                foreach (ValidationResult vr in validationResults)
                {
                    Console.WriteLine(vr.Message);
                }
            }
        }
    }
}
```

Cod: directorul “1. DemoValidation_Customer”. A se urmari testele incluse.

Detalii de lucru

Adaugare de referinte: via Nuget, in Visual Studio, sau prin instalare de MEL separate si adaugare de referinte la Microsoft.Practices.EnterpriseLibrary.Common.dll, Microsoft.Practices.EnterpriseLibrary.Validation.dll (folositi tab-ul

“Browse” pentru a ajunge la locatia de instalare a lui Microsoft Enterprise Library), **System.ComponentModel.DataAnnotations.dll (din tab-ul “.NET”).** Ultimul, desi nu face parte din Enterprise library, este folosit ca auxiliar.

Optional:

Microsoft.Practices.EnterpriseLibrary.Validation.Integration.WinForms.dll,
Microsoft.Practices.EnterpriseLibrary.Validation.Integration.AspNet.dll,
Microsoft.Practices.EnterpriseLibrary.Validation.Integration.WCF.dll.

directive de import:

```
using Microsoft.Practices.EnterpriseLibrary.Validation;
using Microsoft.Practices.EnterpriseLibrary.Validation.Validators;
```

Demo: la clasa.

~~Observatie: pentru un tip “Product” cu proprietatea Price in interior de tip decimal, este imposibil a se face comparatie de tip RangeValidator, deoarece parametrii acestora sunt de tip: int, DateTime, float, double, long, string. Pentru aceasta se sugereaza folosire pattern-ului enterprise “Money”. Ceea ce mai lipseste este comparatia pentru sbyte si short.~~ Pentru tipuri de date ordinale care nu se incadreaza in lista: [float, double, int, long, string, DateTime] se poate folosi varianta in care se transmite tipul dorit - de exemplu typeof(decimal) – iar sub forma de stringuri cele doua limite de validitate: “0”, “100”.

Autovalidarea

Pentru cazuri mai complexe este nevoie de scriere de cod de validare. O clasa poate sa anunte faptul ca se autovalideaza prin folosirea atributului HasSelfValidation la nivel de clasa, iar metodele care fac evaluarea sunt decorate cu atributul SelfValidation.

Exemplu: clasa Product, pentru care validarea insemna o anumita relatie intre preturi sau datele de vanzare/cumparare; aceasta validare nu se poate specifica la nivel de proprietate deoarece o alta proprietate dependenta poate sa nu fie inca cu valoare setata.

Pentru apelarea tuturor validatoarelor unui obiect setat ca valoare a unei proprietati se poate specifica atributul ObjectValidator. Exemplu: clasa Product.

Pentru cazul in care se doreste apelarea tuturor validatoarelor pentru fiecare obiect dintr-o colectie de obiecte, se poate utiliza atributul ObjectCollectionValidator.

O nota aparte pentru validatoarele “Or Composite Validator” si “And Composite Validator” - permit crearea unor expresii bazate pe alte validatoare – a se vedea ExperimentalEntity.

Pentru un obiect de tip ValidationResult proprietatea Key specifica locatia unde a aparut eroarea de validare; daca proprietatea este cu valoare de null, atunci eroarea este la nivel de tip (clasa, structura).

Pentru cazul in care o clasa cu cod de validare este derivata, in cazul in care nu se face suprascriere polimorfica a membrilor atunci se folosesc validarile mostenite; altfel se folosesc validarile implementate prin suprascrieri polimorfice.

Mecanismul de validare poate fi extins sau modificat (sursele sunt date si se pot augmenta /modifica + recompila).

1. Testati daca pentru obiecte care se refera in mod circular (categorie de produse contine lista de produse, fiecare produs are referinta la obiectul categorie asociata) nu cumva apare exceptie la validare (situatie intalnita la MEL versiunea 5). Pentru validare de colectie, folositi ObjectCollectionValidator.

2. (important pentru unit testing) La ce este util parametrul RuleSet pentru metoda `Validation.Validate`? Demonstrati aplicarea selectiva a regulilor de validare.

3. Consultati lista de validatoare din [3], table 1 din cap 6. Cum se face validarea datei de nastere, astfel incat o valoare sa fie acum cel putin 18 ani relativ la momentul rularii?

4. Configurati validatoare folosind consola de configurare din MEL 6, astfel incat regulile de validare sa fie prelucrate din fisier de configurare ([3] cap 6, sectiunea "Configuring Validation Block Rule Sets". Care este utilitatea acestei abordari?

Bibliografie:

[1] http://msdn.microsoft.com/en-us/library/ff648831.aspx

[2] http://www.emoreau.com/Entries/Articles/2009/06/The-Validation-Application-Block.aspx - pentru utilizarea blocului de validare in Windows Forms

[3] Developer's Guide to Microsoft Enterprise Library, cap 6

2. Fluent validator

Folositi Fluent validator, disponibil prin NuGet, pentru a face validari.

Tutorial: https://docs.fluentvalidation.net/en/latest/start.html