
Romanian .NET
User Association
la dispoziția ta din 2004
ronua

WWW.RONUA.RO

Baby Steps in Visual Studio 2010 si .NET 4

Contents

Prefata

In aceasta aplicatie vom arata prin ce trece un programator incepator si ce poate face el usor in Visual Studio 2010 si .NET 4.0 .

Vom face o aplicatie simpla, pe care o vom trece prin mai multi pasi ca sa acoperim (aproape) toate domeniile aplicatiilor ce se pot dezvolta cu .NET.

Vom face abstractie ca majoritatea aplicatiilor pe care le dezvolta un programator sunt deja facute – si ca, de obicei, ce facem sunt imbunatatiri ale vechilor programe. La fiecare episod vi se va da o **tema** * si va invit sa rezolvati **tema** . Actorii principali vor fi Popescu – un programator incepator ce se lupta sa stapineasca VS2010 si Ion – tipul de la vinzari care vine cu idei noi.

Ca sa nu ma dezmint, prima tema va fi sa downloadati Visual Studio Express 2010 de la http://msdn.microsoft.com/express si Sql Server Express

Autori:

(in ordine alfabetica)

Alin Berce, http://alinberce.wordpress.com/ , episodul despre SqlServer

Bogdan Brinzarea, http://bogdanbrinzarea.wordpress.com , episodul F#

Catalin Gheorghiu, http://ronua.ro/CS/blogs/catalin, episoadele Workflow , Window Phone 7 , WebService si /sau WCF

Andrei Rinea, http://blog.andrei.rinea.ro/ , episodul WPF

Melania Danciu, http://melaniadanciu.wordpress.com/ , episodul Silverlight

si cel de pe urma, cu voia dumneavoastra, (desi urasc exprimarea aceasta!)

Andrei Ignat, http://serviciipeweb.ro/iafblog/ , coordonator -restul episoadelor.

Vreti sa participati ? Cititi ce a mai ramas de facut aici !

http://ronua.ro/src=babysteps

Sa inceapa calatoria!

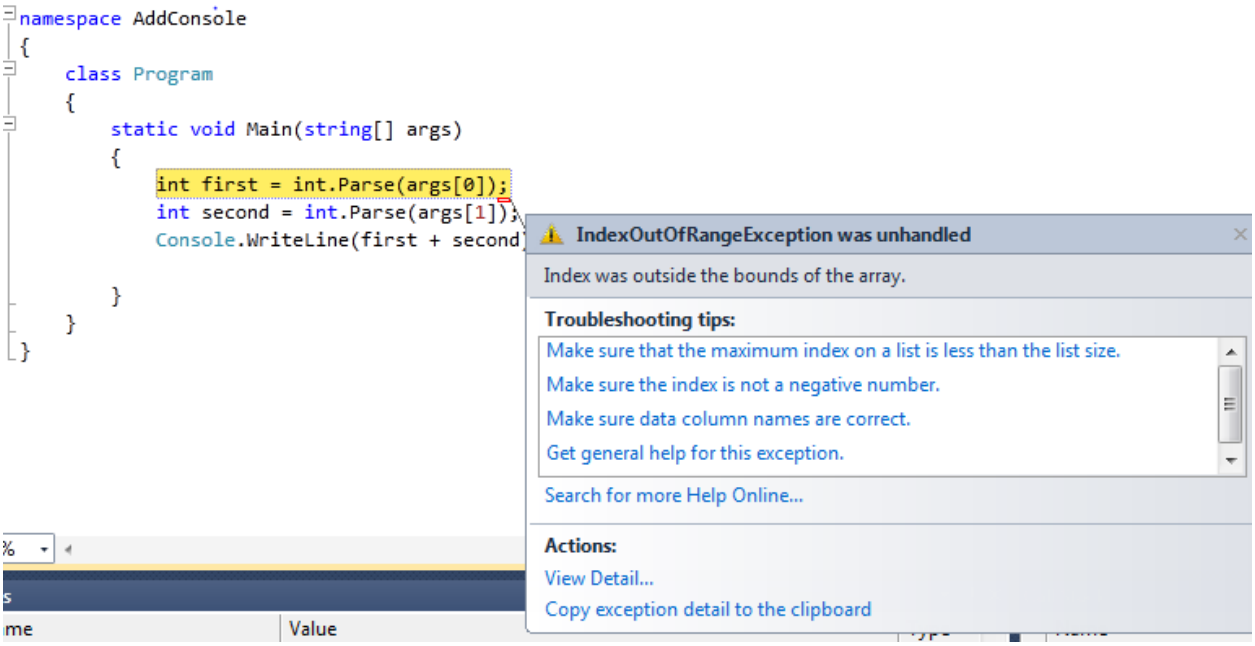
Episodul intii – primul program consola

Popescu, un programator incepator, ajunge la firma unde i se da pe mina VS2010. I se spune sa il studieze si sa invete C#. Peste 3 zile, cind deja se plictisise sa urmeze tutorialele, vine Ion de la vinzari care ii spune sa fie pregatit : va dezvolta prima lui aplicatie. Un client vrea o aplicatie care sa adune doua numere.(nota mea :v-am spus ca e simpla, nu e asa ?)

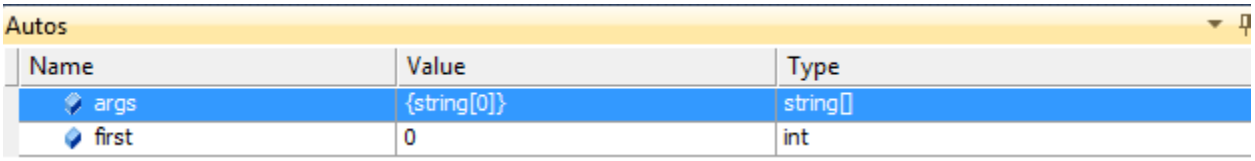
Popescu, nerabdator, nu cere mai multe amanunte, porneste VS2010 si creeaza prima aplicatie Console Application in care aduna primele 2 argumente ale programului :

```
int first = int.Parse(args[0]);
int second = int.Parse(args[1]);
Console.WriteLine(first + second);
```

Cind ruleaza proiectul, ii apare eroarea : IndexOutOfRangeException :



Verifica argumentul args in fereastra de Autos si bineinteles ca este string[0]:



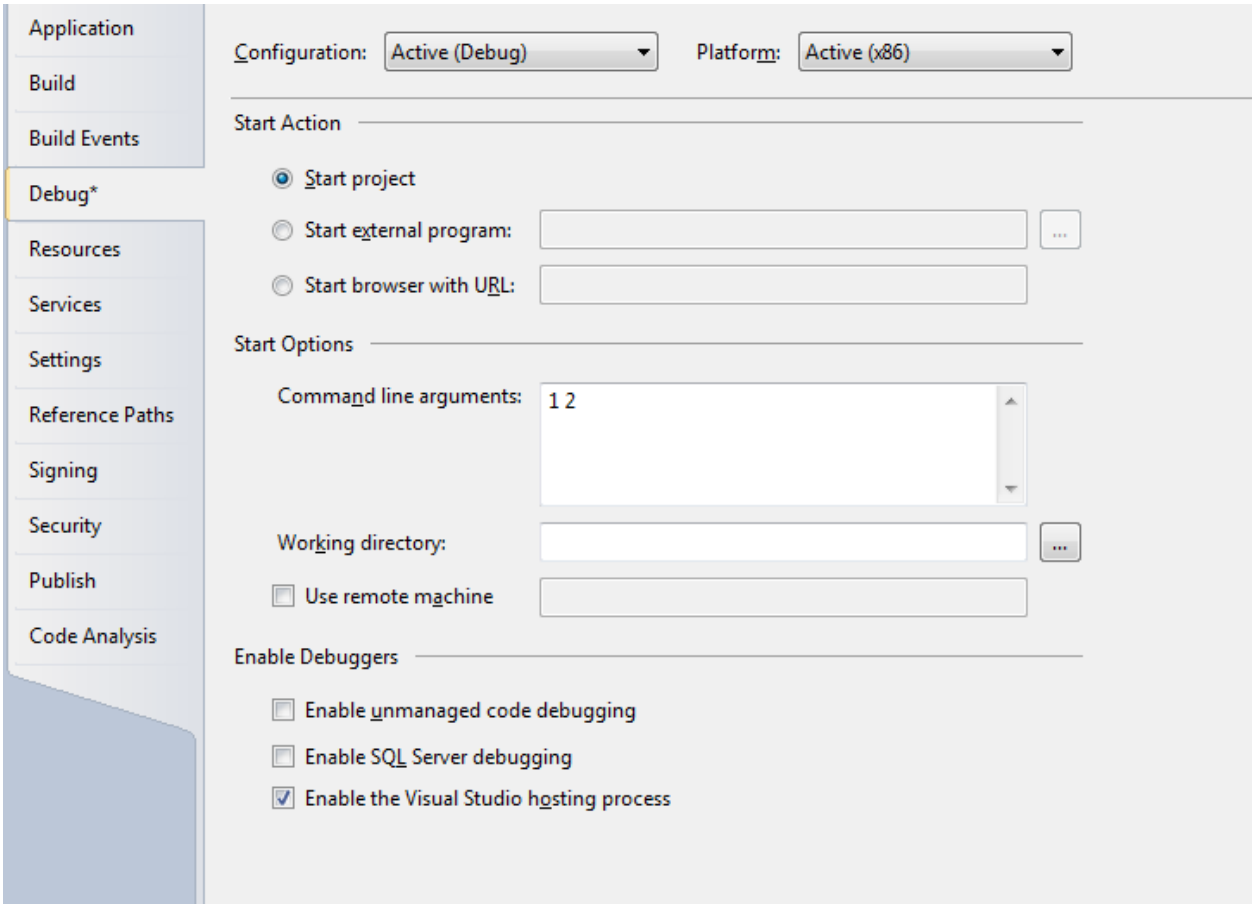
- Normal, isi zice Popescu, doar nu i-am dat nici un parametru. Dar ce fac in acest caz ?

Raspunsul care pare sa fie evident este sa ii dea un mesaj de eroare – programul doar aduna 2 numere, nu ? Asa ajunge la versiunea imbunatatita a programului :

```
 static void Main(string[] args)
        {
            if (args.Length != 2)
            {
                Console.WriteLine("Please enter 2 arguments for this
program");
                return;
            }
            int first = int.Parse(args[0]);
            int second = int.Parse(args[1]);
            Console.WriteLine(first + second);

        }
```

Fericit, ruleaza programul fara argumente -dar fereastra DOS dispare brusc. Pune un breakpoint(F9) pe „return” si programul ii da mesajul scris.

Acum vrea sa ruleze aplicatia cu argumentele 1 si 2 din Visual Studio 2010, asa incit se duce la Project => Properties =>Debug si pune 1 si 2 la Command Line Arguments:



Normal, ii afiseaza 3 la rulare. Ia executabilul, il trimite prin email lui Ion de la vinzari si pleaca acasa fericit – doar a facut primul lui program util in C#!

Exercitiu 1: Cautati intre multele functii int.parse cea care va permite argumente cu separatorul de mii (de ex., in loc de 1 si 2 aveti 1.001(o mie si unu) si 2.003(doua mii trei)) Hint : Apare in episodul urmator.

Exercitiul 2: Ce se intimpla daca in functia Add cele 2 argumente sunt `int.MaxValue si int.MaxValue ?` Ce ati schimba ? Hint : Exista System.Numerics.BigInteger in .NET 4.

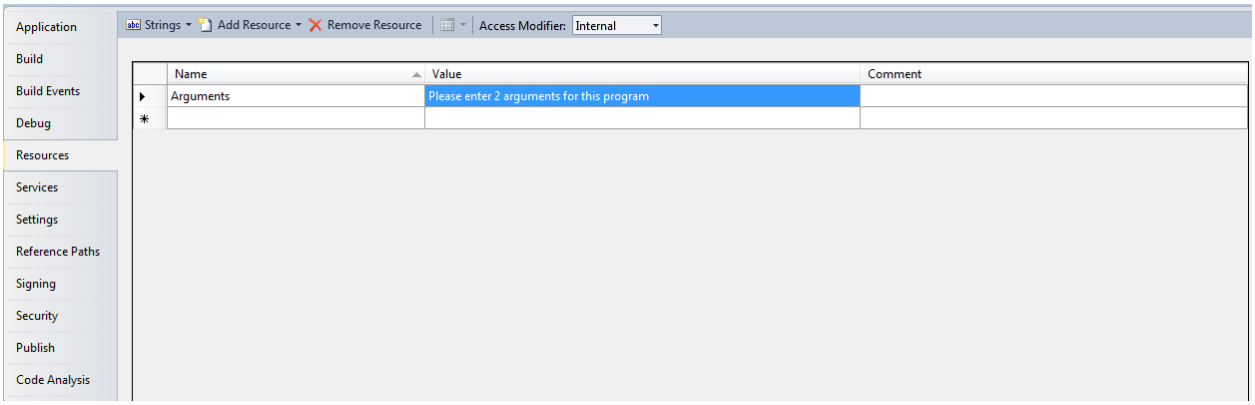
Episodul al doilea– internationalizarea si globalizarea

A doua zi il asteapta un email de „Bravo” – dar si , peste 2 minute, Ion de la vinzari.

Problema este, ii spune el, ca aplicatia vrea sa fie vinduta in SUA si Franta.Mesajul de eroare este OK in engleza – dar nu ar putea , daca PC-ul este in franceza, sa afiseze mesajul in franceza ?

Popescu ii promite ca ii va trimite executabilul chiar azi cu modificarile.

Se duce iar in Project=>Properties=>Debug, sterge cele 2 argumente (1 si 2) ca sa poata verifica mesajul de eroare, cauta tab-ul Resources si creeaza o noua resursa. Pune un nou string, numit „Arguments”, in care valoarea este mesajul de eroare:



A observat ca s-a creeat un Resources.resx si un Resources.Designer.cs – se uita curios in cs si observa :

```
/// <summary>
        ///   Looks up a localized string similar to Please enter 2 arguments
for this program.
        /// </summary>
        internal static string Arguments {
            get {
                return ResourceManager.GetString("Arguments",
resourceCulture);
            }
        }
```

Bucuros, inlocuieste

```
Console.WriteLine("Please enter 2 arguments for this program");
```

Cu

```
Console.WriteLine(AddConsole.Properties.Resources.Arguments);
```

Ruleaza din nou proiectul – si mesajul de eroare i se afiseaza

Acum trebuie sa il creeze in franceza. Copiaza Resources.resx in Resources.fr.resx, sterge codul din designer si inlocuieste in Resources.fr.resx pe "Please enter 2 arguments for this program" cu „Je vous en prie de passer deux arguments"

Ca sa verifice, prima linie din program va fi :

```
System.Threading.Thread.CurrentThread.CurrentUICulture= new
System.Globalization.CultureInfo("fr-FR");
```

Ruleaza programul, ii afiseaza „Je vous en prie de passer deux arguments”, scoate linia respectiva cu franceza, compileaza din nou.

T otusi, se intreaba el, daca ar fi fost sa am mesajul de eroare in franceza si numerele in alt format, de exemplu in Romana, unde 1234 se scrie 1.234 – cum as fi facut ?

Mai intii, isi spune el, trebuie sa stiu cine e „Romana” – si scrie urmatorul cod:

```
CultureInfo ciRO = new CultureInfo("ro-RO");
```

In al doilea rind, trebuie sa fie afectat int.Parse – si, uitindu-se la definitiile lui int.parse, gaseste :

```
int.Parse(args[0],System.Globalization.NumberStyles.AllowThousands,ciRO);
```

In acest moment daca porneste programul cu argumentele 1.450 3.456 , rezultatul este:

4906

OK, isi zice Popescu, pot sa am mesajele de eroare in franceza(CurrentUICulture) si interpretarea cifrelor in alt format(ciRO).

Tema pentru acasa:

1. Creati o noua resursa in Spaniola, Germana ,Italiana sau Klingoniana .Verificati!
2. Studiati cazul Elvetiei, care are 3 limbi oficiale(Germana, Franceza si Italiana). Cum ati da mesajele de eroare pentru un elvetian ?

Episodul al treilea –refactorizare si alte tipuri de proiecte

Peste o luna la Popescu se iveste Ion de la vinzari care ii spune:

- Programul este superb , dar ... avem doua oportunitati nevalorificate
- Da? , intreaba Popescu.
- Da. Sunt citiva clienti care nu vor sa aiba o consola dos urita,ci sa aiba un program cu butoane si casute unde sa puna cifrele. Si inca citiva care au Linux. Poti face ceva pentru ei ?
- Pentru cei cu butoane e Ok. Dar pentru cei cu linux – e ok daca facem un site?
- Da, raspunde cel de la vinzari.
- OK, ma apuc!

Mai intii Popescu isi spune ca proiectul cu butoane e mai simplu. Asa incit adauga un Windows Forms project la solutia existenta, pune 2 textbox-uri pe el si doua butoane(Add si Exit) si pe evenimentul de Click pe add se surprinde scriind (aproape) acelasi cod ca pe consola :

```
public partial class frmAdd : Form
    {
        public frmAdd()
        {
            InitializeComponent();
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            int first = int.Parse(txtFirst.Text);
            int second = int.Parse(txtSecond.Text);
            MessageBox.Show(this,""+first + second);

        }
    }
```

Popescu stie ca unul din Anti-Patterns(http://en.wikipedia.org/wiki/Antipatterns) este copy and paste(http://en.wikipedia.org/wiki/Copy_and_paste_programming) - asa ca se gindeste cum sa faca sa aiba codul comun in acelasi loc.

Raspunsul care ii vine este : un dll(proiect de tipul Class Library) care sa contina codul respectiv. Ar trebui sa fie o clasa care sa contina functia add si sa stie sa adune doua stringuri . Mai trebuie sa poata si parsa stringurile in functie de separatorul de mii .Asa ajunge la urmatoarea clasa:

```
public class clsAdd
    {
        private CultureInfo CI;

        public clsAdd(CultureInfo ci)
        {
            CI = ci;
        }
        public clsAdd()
            : this(Thread.CurrentThread.CurrentCulture)
        {
        }

        public int Add(string first, string second)
        {
            int f = int.Parse(first, NumberStyles.AllowThousands,
CI.NumberFormat);
            int s = int.Parse(second, NumberStyles.AllowThousands,
CI.NumberFormat);
            return f + s;
        }
    }
```

Acum adauga la referintele celor 2 proiecte proiectul acesta si rescrie codul:

Pentru Windows Forms:

```
            clsAdd add = new clsAdd();
            MessageBox.Show(this, "" + add.Add(txtFirst.Text,
txtSecond.Text));
```

Pentru consola:

```
clsAdd add = new clsAdd();

            Console.WriteLine(add.Add(args[0],args[1]));
```

Acum nu ii ramine decit sa verifice ce a facut – si isi da seama, ca ,de fapt, tot ceea ce are de facut este sa verifice functia Add din clsAdd.

Se gindeste el daca nu ar fi bine sa faca un test automat din asta

Adauga un proiect de tipul „Test Project” , adauga referinta la proiectul „AddObjects” si scrie rapid un cod de verificare:

```
[TestMethod]
        public void TestAdd()
        {
            clsAdd c = new clsAdd();
            int result = c.Add("1", "2");
            Assert.AreEqual(3, result, "The result must be three");
        }
```

Acum stie ca nu mai are nevoie sa verifice codul din Windows Forms sau Consola – ci doar sa verifice automat , prin rularea proiectului de test, functia TestAdd.

Ruleaza proiectul de test si vede fericit ca testul e „passed”

Exercitiu 1: Construiti un proiect WPF in care folositi aceeasi clasa, clsAdd

Exercitiu 2:Construiti un proiect Windows Mobile in care folositi aceeasi clasa , clsAdd

Exercitiu 3: Construiti un proiect de setup pentru proiectul consola si un clickonce pentru proiectul Winforms

Exercitiu 4 : Folositi un Source Version Control (Team Foundation Server, SVN, CVS, GIT). Incercati sa implementati un continous integration system.

Episodul al patrulea–Generics si dynamic

Dimineata pe Popescu il asteapta Ion de la vinzari care il felicita pentru nemaipomenitul lui program

Ion : Am citiva clienti care ar fi interesati de functia respectiva, Add... dar ar vrea sa o foloseasca ei fara sa mai foloseasca consola noastra. Se poate ?

Popescu : Sigur ca da... atita timp cit folosesc .NET. Ei ce folosesc ?

Ion : Cred ca .NET .O sa ma interesez Poti sa imi scrii cum sa il foloseasca ?

Popescu : Da. Altceva?

Ion : Altii au cerut ca programul sa adune si doua numere zecimale,se poate face usor ?

Popescu : O sa incerc. Altceva ?

Ion : Deocamdata nimic. Poti sa o faci ?

Popescu : Da, incerc acum.

Popescu revine la masa de lucru si ii trimite instructiuni complete lui Ion despre cum poate cineva folosi dll-ul AddConsole si ii trimite si un exemplu de functionare(practic prima aplicatie consola)

Acum se gindeste cum ar face el functia ce aduna doua zecimale. Ar adauga o functie ce face parsing la doua zecimale, dar ar trebui sa faca apoi si pentru long, short si asa mai departe – si nu ii convine sa munceasca in plus. Gindindu-se un pic mai mult, isi da seama ca functiile respective difera doar prin tipul argumentelor – asa ca ajunge natural la Generics.

Prima varianta a fost o noua functie, AddNew , care incerca sa faca adunarea a doua „value-types”:

```
public T AddNew<T>(string first, string second)
            where T : struct
        {
            Type t=typeof(T);
            T f = (T)Convert.ChangeType(first, t, CI.NumberFormat);
            T s= (T)Convert.ChangeType(second, t, CI.NumberFormat);
            T ret = f + s ;
```

```
            return ret;
        }
```

Problema care aparea era ca f si s nu puteau fi adunate – deoarece nu stiau daca exista operatorul + intre ele . Asa ca , stiind ca VS2010 vine cu dynamic, se hotaraste sa rescrie in dynamic :

```
public T AddNew<T>(string first, string second)
            where T : struct
        {
            Type t=typeof(T);
            dynamic f = Convert.ChangeType(first, t, CI.NumberFormat);
            dynamic s = Convert.ChangeType(second, t, CI.NumberFormat);
            dynamic ret = f + s ;

            return (T)ret;
        }
```

Avind deja proiectul de test pregatit adauga un test cu double si inca unul cu int:

```
[TestMethod]
        public void TestAddNew()
        {
            clsAdd c = new clsAdd();
            Assert.AreEqual(3, c.AddNew<int>("1", "2"), "The result must be
three");
            Assert.AreEqual(3.5, c.AddNew<double>("1.4", "2.1"), "The result
must be three and a half");

        }
```

Ruleaza proiectul de test si vede fericit ca toatele testele sunt „passed”

PS : Pentru studierea cum s-ar fi putut face fara dynamic, studiati atent http://groups.google.com/group/microsoft.public.dotnet.languages.csharp/msg/2f5c2dd862020759 si http://www.codeproject.com/KB/cs/genericnumerics.aspx

Exercitiu 1: Cum ati modifica proiectul consola ca sa ia in considerare daca sa foloseasca Add sau AddNew ? Hint : vedeti separatorul curent de zecimale

Exercitiul 2: Cum ati modifica proiectul Winforms ca sa ia in considerare daca sa foloseasca Add sau AddNew ? Hint : Aveti posibilitatea de a schimba setarile de limba pentru aplicatie(In nici un caz pentru user!). Nu uitati cazul Elvetiei.

Episodul al cincilea : Argumente optionale si expresii lambda(functii)

A doua zi la Popescu revine Ion, tipul de la vinzari:

Ion: Clientii la care am trimis proiectul cu AddNew sunt fericiti. Dar citiva se pling ca nu reusesc sa il foloseasca din Excel. Poti sa ii rezolvi ?

Popescu : Desigur. Altceva ?

Ion : E ceva mai ciudat. Citiva clienti mai vechi ce au doar dll-ul ar vrea ca sa foloseasca functia ta , add, si pentru scaderi. Adica cind vor ei sa adune si cind vor ei sa scada. Poti sa o faci ?

Popescu : Desigur. Altceva ?

Ion:Acum, ca ma intrebi, citiva din ei au zis ceva de genul ca nu vor neaparat adunare, ci sa iti dea ei functia. Se poate si asta?

Popescu : Desigur. Altceva ?

Ion : Deocamdata atit. E gata azi ?

Popescu: Desigur, la sfirsitul zilei.

Popescu se apuca de primul task, deoarece i se pare mai usor. Stie ca din VS2010 exista argumente optionale in C#, asa ca modifica functia Add:

```
public int Add(string first, string second, bool difference=false)
        {
            int f = int.Parse(first, NumberStyles.AllowThousands,
CI.NumberFormat);
            int s = int.Parse(second, NumberStyles.AllowThousands,
CI.NumberFormat);
            if (difference)
                return f - s;
            else
                return f + s;
        }
```

Pentru ca isi facuse testele, le ruleaza sa vada daca nu este vreo problema. Nu este, asa ca scrie un test si pentru noul argument optional :

```
[TestMethod]
        public void TestDifference()
        {
            clsAdd c = new clsAdd();
            int result = c.Add("2", "1",true);
            Assert.AreEqual(1, result, "The result must be one");
        }
```

Ruleaza proiectul de test si acum vede 3 teste „verzi”.

Pentru a doua cerere rescrie functia Add ca sa poata sa ii fie trimisa ce functie vor clientii:

```
public int Add(string first, string second, Func<int,int,int> func)
        {
            int f = int.Parse(first, NumberStyles.AllowThousands,
CI.NumberFormat);
            int s = int.Parse(second, NumberStyles.AllowThousands,
CI.NumberFormat);
            return func(f, s);
        }
```

Si scrie repede si un test:

```
[TestMethod]
        public void TestMultiply()
        {
            clsAdd c = new clsAdd();
            Assert.AreEqual(6, c.Add("2", "3",(x,y)=>{return x*y;}), "The
result must be six");

        }
```

Ruleaza proiectul de test si acum vede 4 teste „verzi”.

Exercitiul 1: Chemati functia Add cu impartire in loc de inmultire. Ce observati ?

Exercitiul 2: Daca vi s-ar fi spus ca un client vrea pentru functia add sa intoarceti 0 daca numarul e divizibil cu 2 si 1 daca nu e divizibil, ce ati face ? Hint : puteti folosi functia func .

Episodul al saselea– use case si internet (inregistrarea clientilor si asp.net)

La Popescu vine iarasi Ion, cel de la vinzari, cu o idee noua : ce ar fi daca am expune fnctionalitatea pe internet ? In acest fel, motiveaza Ion, am putea avea un target intr-adevar global :ne-ar putea veni clienti de oriunde. Popescu ii raspunde ca „nimic mai simplu!” - dar are citeva intrebari :

- Draga Ion, cum facem cu identificarea clientilor ? E de ajuns email si parola ?
- Da, sunt de acord.
- Si cu plata cum facem ?

- Pai plata o fac si o trimit prin fax, apoi cineva va intra in program si va identifica clientul si ii va da o perioada de gratie.
- Aha ...

E clar pentru Popescu ca acum e vorba despre un site adevarat , deci va trebui sa faca citeva UseCase-uri (http://en.wikipedia.org/wiki/Use_case) .

UC1 - inregistrare

Scop : inregistrarea utilizatorului pentru a beneficia de un demo al aplicatiei

Actori: Utilizator neinregistrat

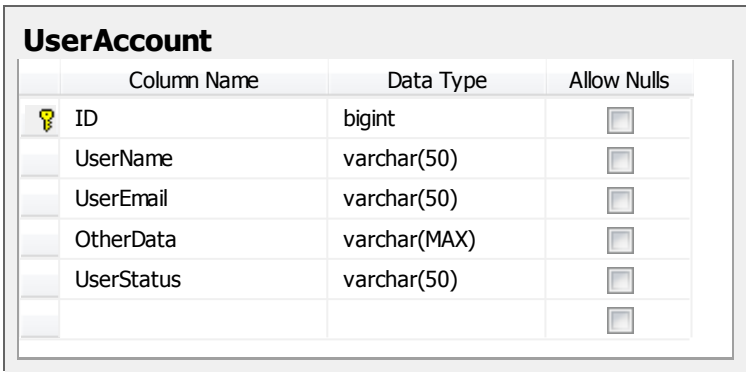
Functionalitate : Utilizatorul se va duce pe www.<site>.com si va dori sa se inregistreze pentru a beneficia de un demo al aplicatiei cu o durata de 1 luna. Inregistrarea va fi facuta conform standardelor „double-opt-in" .

Scrie acest UC pe email si vrea sa il trimita celui de la vinzari spre revizuire. Totusi, isi zice Popescu, nu ar fi mai simplu sa fie undeva in proiect – decit pe email ? Isi aduce aminte ca exista un tip de proiect – de tipul „Modeling Project" – si ii acorda o sansa citind documentatia de la adresa http://msdn.microsoft.com/en-us/library/dd409427.aspx . Incurajat, face un proiect de tipul modeling project, adauga un nou item de tipul use case , adauga actori si actiuni si vrea sa trimita acum celui de la vinzari pentru aprobare. Dar cum sa trimita in email ? Isi aduce aminte de universalele CTRL+A (select all) si (CTRL+C cu CTRL +V) si le foloseste – iata ce ii rezulta :

Register
with username
and email

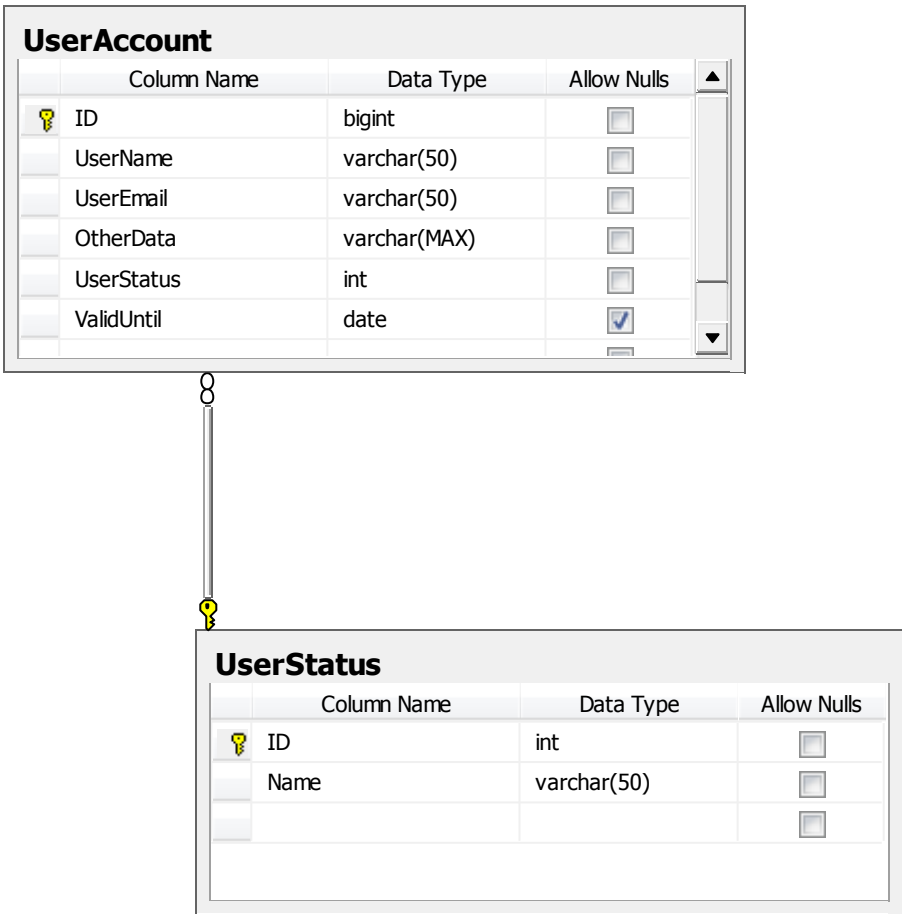
Trimite celui de la vinzari si incepe lucrul pentru site-ul ASP.NET . Vede ca exista doua tipuri de site-uri – ASP.NET si ASP.NET MVC. Incearca sa citeasca documentatia de la MVC si promite ca va incerca ... mai tirziu. Porneste cu constructia Bazei de date – porneste in stil clasic, de la SQL Server Management

Studio , creeaza o noua baza de date si adauga o nou „Database diagram”.Acolo creeaza o noua tabela :



In „UserStatus” o sa puna status-ul curent al user-ului(neinregistrat, inregistrat) iar in „Other Data” va pune alte detalii despre inregistrarea user-ului(IP-ul, de exemplu).

Citind despre Formele normale ale bazelor de date(http://en.wikipedia.org/wiki/Database_normalization#Normal_forms) , isi da seama ca acea coloana, UserStatus, ar trebui pusa intr-o tabela independenta - asa ca reface baza de date – si cu ocazia asta adauga si un „ValidUntil” pentru user-ul respectiv :



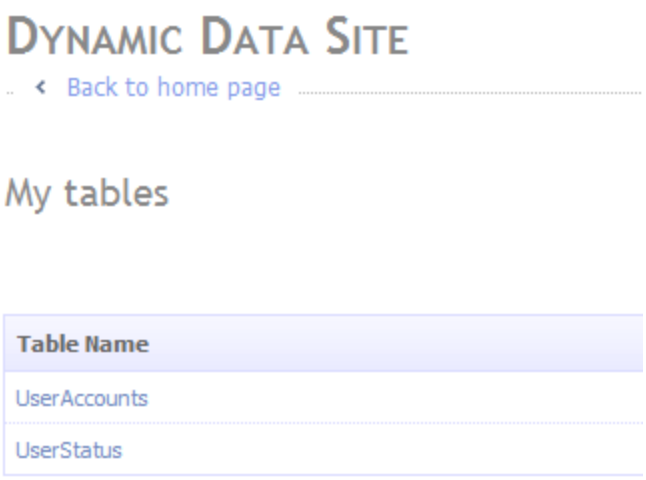
(Paranteza : cind vrea sa salveze, nu poate. Asta din cauza ca trebuie sa se duca in Tools=>Options=>Designers=>Prevent Save Changes that requires table re-creation)

Acum ce ar trebui sa mai faca este o interfata de administrare pentru aceste doua tabele – astfel incit cineva sa poata edita acel „ValidUntil” sau sa stearga user-ii neinregistrati . Adauga un proiect de tipul „Dynamic Data Entities WebSite” , ii adauga un item „ADO.NET Entity Model” facind sa ia tabelele din baza de date si modifica linia din global.asax:

```
DefaultModel.RegisterContext(typeof(BabyStepsModel.BabyStepsEntities), new
ContextConfiguration() { ScaffoldAllTables = true });
```

Ruleaza proiectul si observa deja ca are tabelele facute pentru partea de administrare :

http://ronua.ro/src=babysteps



De acum nu mai are de facut decit 2 parti :

1. Aplicatia propriu –zisa (implica o aplicatie Web aproape identica cu cea de windows forms)
2. Identificarea user-ului curent ca e valid(inregistrat si „validuntil”< now)

Tema :

Exercitiul 1 : Continuati aplicatia cu cele 2 parti. Daca nu aveti Sql Server , downloadati versiunea Express de la (http://www.microsoft.com/express/)

Exercitiul 2: faceti un UC pentru „validUntil” .

Exercitiul 3: cum ati modifica baza de date daca vi s-ar cere sa aveti un raport cu „intirzierile” utilizatorului de la plata ?

Exercitiul 4 : Studiati ASP.NET MVC. Da, e mai complicat – dar e mai rapid si mai usor de configurat.

Episodul al saptelea –deschiderea aplicatiei internet catre altii WebService si /sau WCF

Intr-o dimineata primeste un telefon de la Ion : Ma aflu la un client care are Linux pe masina. Putem sa ii facem un demo de cum functioneaza aplicatia ?

Desigur, raspunde Popescu, avem site-ul web. Sa se inregistreze si gata!

Scuze... ar vrea sa foloseasca aplicatia in PHP. Se poate ?

Lasa-ma cinci minute, spune Popescu.

Studiind problema, vede ca exista un modul de PHP care interactioneaza cu WebService(http://php.net/manual/en/book.soap.php) . Asa incit la aplicatia Web adauga un WebService in care replica functia Add (o mai tineti minte ? Prima functie din consola !) facind apel la dll-ul deja existent. Ii trimite adresa lui Ion precum si cum se poate folosi din PHP. Dar stie ca WebService este demodat – si ca ar putea folosi WCF.

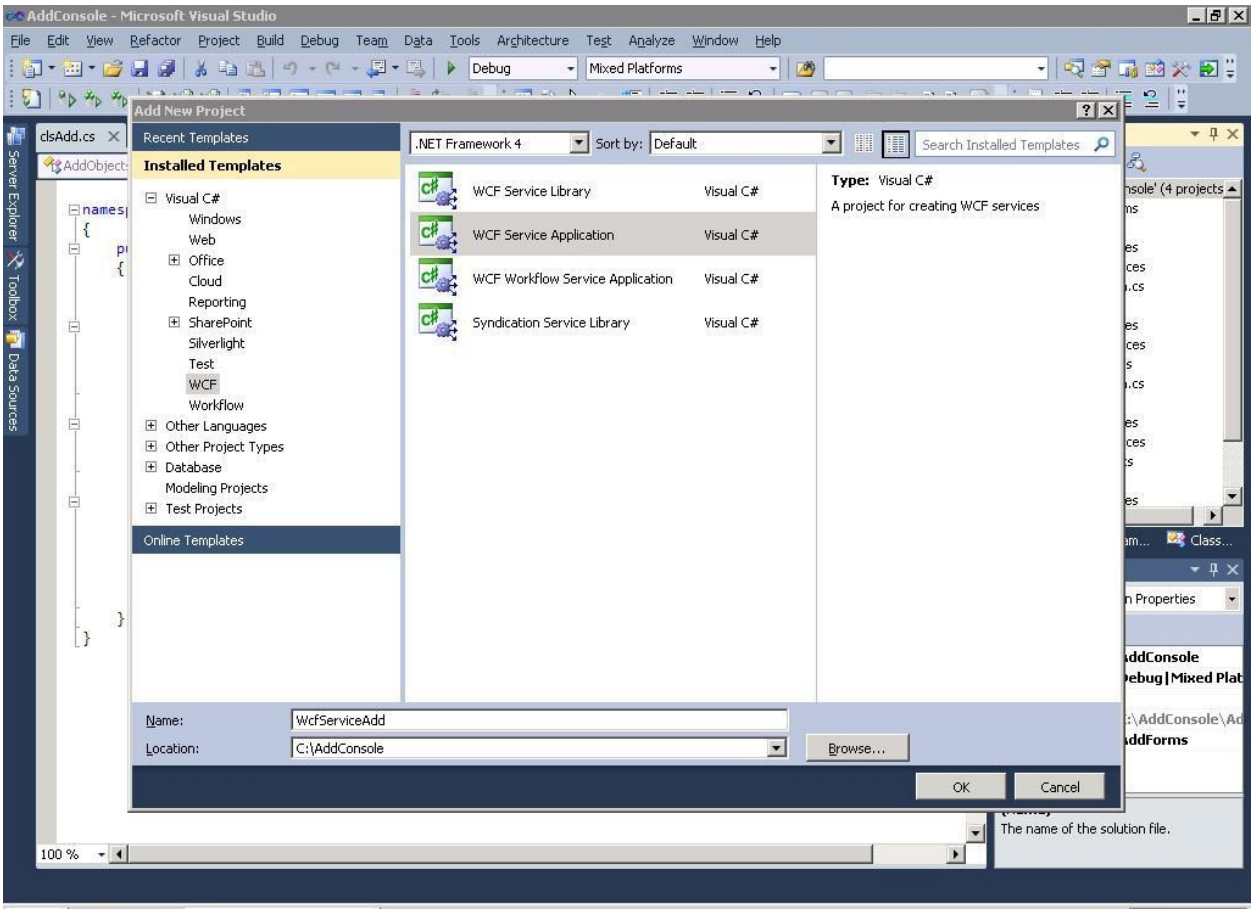
Avantajele WCF versus WebService, cea mai de impact (si grafica) reprezentare am gasit-o aici (http://keithelder.net/blog/archive/2008/10/17/WCF-vs-ASMX-WebServices.aspx)

vs

http://ronua.ro/src=babysteps

Dar cum a facut totusi Popescu un serviciu Windows Comunication Foundation?
Normal a deschis Visual Studio 2010 si a facut un nou proiect WCF Service

Application:

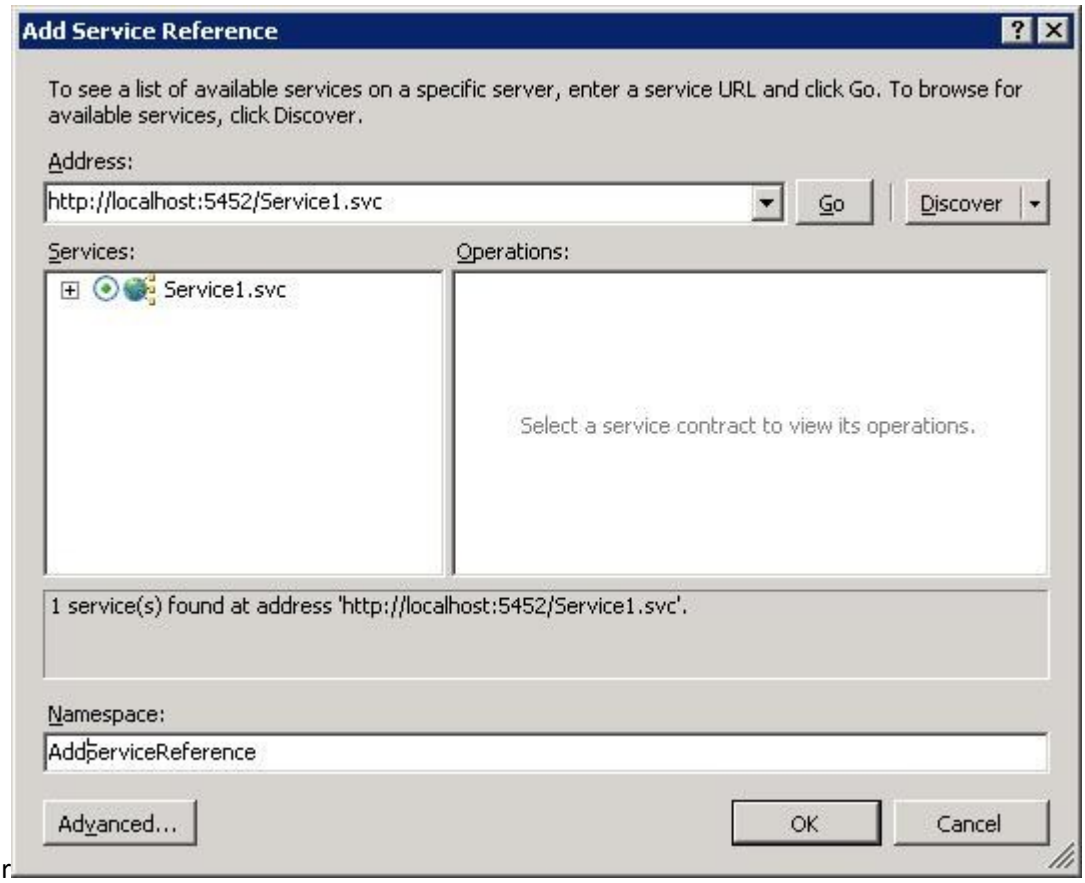


Dupa ce s-a generat proiectul sa vada si cum sa faca operatiile...

Pune o referinta la dll care implementeaza adunarea si apoi face click dreapta pe fisierul Service1.svc si alege View Code, si scrie o metoda care apleaza metoda de adunare.

```
using AddObjects;
...
public int Add(string first, string second)
        {
            return new clsAdd().Add(first, second);
        }
```

Si acuma modifica si aplicatia client si ca sa vedem ca merge, ne ducem la proiectul AddForms si click dreapta pe Reference, Add Service Reference,



Discover

Si daca face click pe service contract nu gaseste metoda de adunare aduagta

Ion isi da seama ca noua metoda nu este expusa de seviciul WCF. Si deci merge in IService1.cs si adauga urmatorul Operation Contract

```
[OperationContract]
        int Add(string first, string second);
```

La Discover totul e OK acuma, si deci trece la modificari in codul Form, scoate referintele laAddObjects si in metoda btnAdd_Click inlocuieste `codul existent cu` urmatorul cod.

```
AddServiceReference.Service1Client addServiceClient = new
AddServiceReference.Service1Client();
MessageBox.Show(this, "" + addServiceClient.Add(txtFirst.Text,
txtSecond.Text));
addServiceClient.Close();
```

http://ronua.ro/src=babysteps

Porneste o rulare in debugger (si serviciul WCF si AddForms fac parte din acceasi solutie si AddForms este setat ca proiectul lansat la debugging), si totul merge la fel ca inainte, desi acuma functionalitatea este in „cloud” ☺.

Exercitiul 1 : Construiti efectiv acel WebService

Exercitiul 2 : Gasiti un WebService si folositi-l(de ex., il puteti folosi pe al meu de la http://infovalutar.ro/curs.asmx)

Exercitiul 3 : In acest studiu de caz, folosirea functiei Add se plateste(inclusiv pe Web)

Ce ar trebui sa faca Popescu ca numai un user inregistrat toti sa poata folosi acest serviciu ? Hint : EnableSession

Episodul al optulea - deschiderea catre lumea Office si COM.

Din nou vine Ion si aduce vorba despre un client, ce vrea sa foloseasca aplicatia lui din Excel. Utilizatorul stie doar VBA si nu are habar de .NET. Ar putea Popescu sa il ajute ?

Sigur ca da, raspunde Popescu.

Ion : - Ah, si inca ceva. Un system admin care stie doar VBScript ar vrea si el sa testeze . E OK ?

Sigur ca da, raspunde Popescu, stiind ca va face un singur cod pentru amindoua.

– si se apuca sa studieze problema.Vede ca exista atributele ComImport si Guid pentru asa ceva - si bucuros , le adauga la clsAdd (a generat un nou Guid din Tools=>GenerateNewGuid)

```
[ComImport]
    [Guid("C1332946-2314-4d7f-A259-DEBAB9202581")]
    public class clsAdd
    {
private CultureInfo CI;


        public clsAdd(CultureInfo ci)
        {
            CI = ci;
        }
        public clsAdd()
            : this(Thread.CurrentThread.CurrentCulture)
```

```
        {
        }
```

Dar – surpriza! Eroarea pe care i-o da compilarea este:

```
Error  2     A class with the ComImport attribute cannot have a user-defined
constructor
```

Normal ca nu vrea sa renunte la constructorii lui – asa incit va trebui sa isi defineasca o interfata pentru asta :

```
InterfaceType(ComInterfaceType.InterfaceIsDual)]
    [Guid("C1332946-2314-4d7f-A259-DEBAB9202581")]
    public interface IAdd
    {
        int Add(string first, string second, bool difference=false);
    }

    [Guid("507FD7ED-5A8E-475d-900C-737E8A74E978")]
    [ClassInterface(ClassInterfaceType.None)]
    [ProgId("AddObjects.Add")]
    public class clsAdd : IAdd
    {
```

Acum se duce in Command tools (rulat ca administrator) si executa

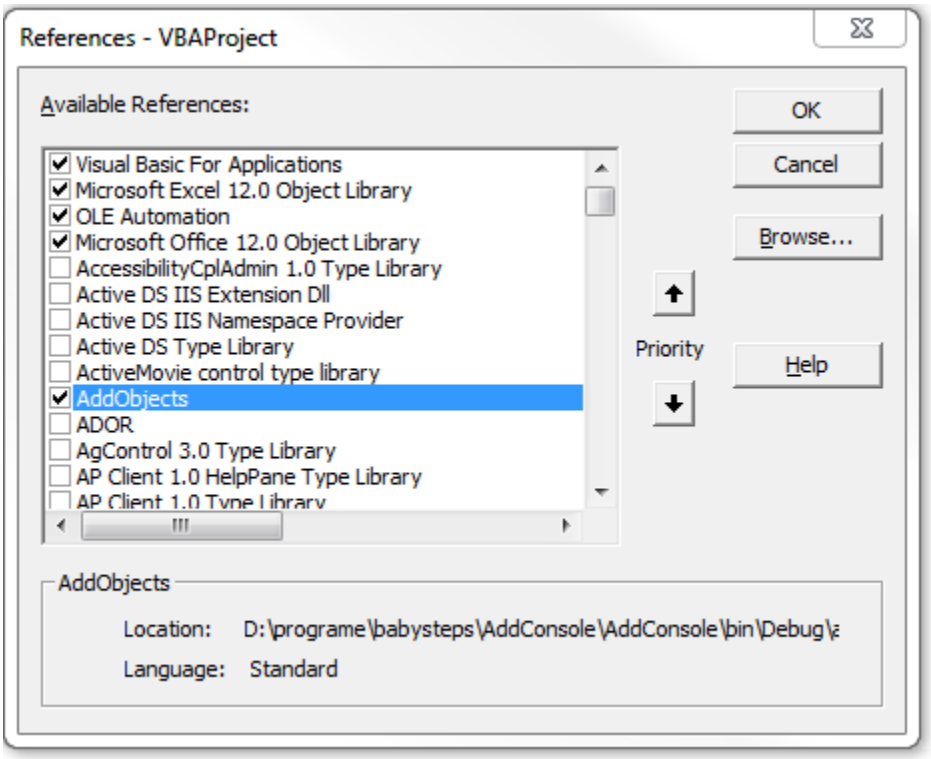
```
regasm addobjects.dll /tlb addobjects.tlb
```

Outputul este:

```
Microsoft (R) .NET Framework Assembly Registration Utility 4.0.21006.1
Copyright (C) Microsoft Corporation 1998-2004.  All rights reserved.

Assembly exported to [...]\bin\Debug\addobjects.tlb', and the type library was
registered successfully
```

Se duce in Excel, apasa ALT + F11 ca sa apara VBA ,se duce in Tools=> References si selectioneaza libraria lui :

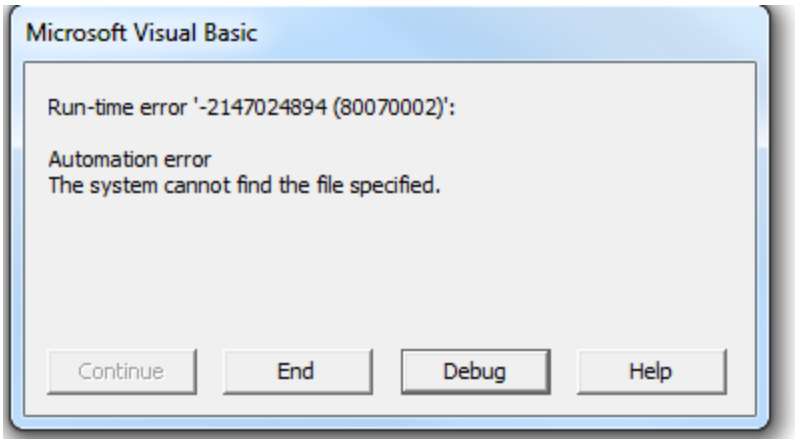
http://ronua.ro/src=babysteps



Dupa care scrie codul :

```
Option Explicit

Sub x()
    Dim a As AddObjects.clsAdd
    Set a = New AddObjects.clsAdd
    MsgBox a.Add("1", "2")
End Sub
```

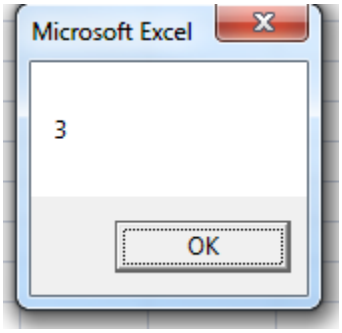
Din pacate, ii da un alt mesaj de eroare:



Citind in plus, isi da seama ca trebuie sa isi genereze un StrongName pentru acest dll ... si deci mai face o setare in Project=> Properties =>Signing selectind SignTheAssembly

Compileaza si, ca sa ii fie mai usor, selecteaza in Project=>Properties=>Build setarea „Register For Com Interop”

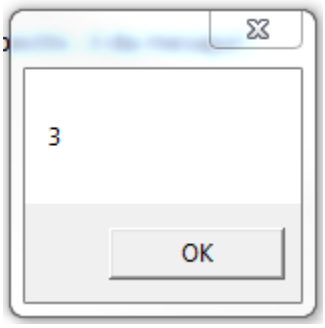
Revine la Excel si ruleaza cu F5 codul respectiv . Ii da mesajul :



In fine , pentru ssytem admin scrie un fisier a.vbs cu urmatorul cod :

```
dim u
set u=CreateObject("AddObjects.Add")
Msgbox u.Add("1","2")
```

Il executa – si ii da acelasi mesaj minunat:



In acest moment, pentru a face deployment la client nu mai sunt decit citiva pasi – pe care ii regasiti in exercitii.

Exercitiul 1: folositi dll-ul in Word.

Exercitiul 2 : cum ati face la client ? Hint : Regasm

Exercitiul 3: Modificati proiectul de setup astfel incit sa peremita deploymentul acestui dll ca si COM – enabled. Hint : vsdrfCom

Episodul al noualea– F#

Citind pe Internet, Popescu afla ca odata cu Visual Studio 2010 a fost introdus un nou limbaj : F#. Decis sa detina si acest as in maneca pentru a-i putea raspunde prompt lui Ion pentru cerinte viitoare, Popescu se decide sa vada cum decurg lucrurile in acest nou limbaj.

Deschide Visual Studio 2010 si creaza un nou proiect de tipul Visual F# ->Windows -> F# Application.

Citise ca spre deosebire de celelalte limbaje din .NET, F# permite sa scrii cod si sa-l testezi fara a trebui sa rulezi in modul clasic aplicatia. Deschide fisierul Script.fsx si scrie primele linii de cod in F#:

```
let add x y = x + y

printfn "%A" (add 1 2)
```

Selectand codul si apasand combinatia ALT + ENTER obtine in fereastra F# Interactive urmatoarele:

```
>
3

val add : int -> int -> int
```

Foarte interesant!! A selectat codul pe care a dorit sa-l ruleze si pur si simplu s-a executat! In plus compilatorul a dedus semnatura functiei direct din codul pe care l-a scris! Nemaipomenit!

Cum ar fi daca ar stoca mai intai rezultatul adunarii intr-o “variabila”

```
let add x y = x + y

let result = add 1 2

printfn "%A" result
```

Popescu citise ca F# este un limbaj functional si printre lucrurile diferite se numara si faptul ca functiile si obiectele sunt similare.

Se gandeste ca i-ar fi utila si o functie care sa adauge 10 peste valorile primite ca parametru.

```
let add10 x y = x + y + 10
```

Uitandu-se atent observa ca semnatura functiei asa cum o detecteaza compilatorul este cea pentru numere intregi chiar daca functia nu este folosita deloc. Compilatorul detecteaza semnatura valabila a functiei pornind doar de la valoarea intreaga 10.

```
val add10 : int -> int -> int
```

Vazand acest lucru, Popescu incearca sa apeleze functia add si pentru 2 numere float.

```
let add x y = x + y

printfn "%A" (add 1 2)

let add10 x y = x + 10 + y

printfn "%A" (add 1.0 2.0)
```

Vede ca Intellisense-ul ii semnaleaza o incompatibilitate intre parametrii cu care este apelata functia a doua oara. Incearca sa ruleze totusi codul folosind combinatia ALT + ENTER si obtine urmatoarea eroare in fereastra interactiva.

```
Script.fsx(10,19): error FS0001: This expression was expected to have type
    int
but here has type
    float
```

Hmm.. Se pare ca totusi compilatorul vede ca incearca sa folosesc functia cu parametrii de tip diferit si nu functioneaza cum trebuie. Se gandeste sa comenteze primul apel al functiei.

```
let add x y = x + y

// printfn "%A" (add 1 2)

let add10 x y = x + 10 + y

printfn "%A" (add 1.0 2.0)
```

Nu mai obtine nicio eroare si ruland exemplu obtine rezultatul dorit: 3.0.

Functia add poate fi folosita cu mai multe tipuri de parametri dar de fiecare data numai cu acelasi tip de parametri .

Se gandeste si decide ca functia sa nu fie apelata decat cu parametri intregi.

```
let add (x:int) (y:int) = x + y
```

Gandidu-se la exemplul precedent in care a primit ca parametru o functie si gandidu-se ca in F# o functie este aproximativ ca o variabila, il rescrie astfel

```
let add x y f = f x y
printfn "%A" (add 2 3 (fun a b -> a * b))
```

Un prieten matematician ii vorbise despre functii partiale si s-a gandit sa rescrie functia de adunare a doua numere astfel

```
let add x y = x + y
let add1 = add 1
printfn "%A" (add1 2)
```

De asemenea, o alta functionalitate interesanta de care auzisese este cea a pipeline-urilor (duble in cazul nostru)

```
let add x y = x + y

printfn "%A" ( (1,2) ||> add)
```

Unul dintre lucrurile care ii placusera lui Popescu cand citise despre F# erau multiplele functii ajutatoare disponibile in majoritatea structurilor de date din limbaj.

De exemplu, pentru a aduna aceleasi 2 numere intr-un array si folosind pipeline-urile, scrie urmatorul cod:

```
printfn "%A" ( (1,2) ||> add)
Similar stau lucrurile si pentru liste si secvente
printfn "%A" ( [1;2] |> List.sum)
printfn "%A" ( {1..2} |> Seq.sum)
```

Gandindu-se ca poate va trebui sa adune si mai multe numere, Popescu a fost extrem de incantat ca programul lui nu ar trebui sa se schimbe de tot si ca intr-o linie de cod poate rezolva cerinta fara a folosi variabile ajutatoare.

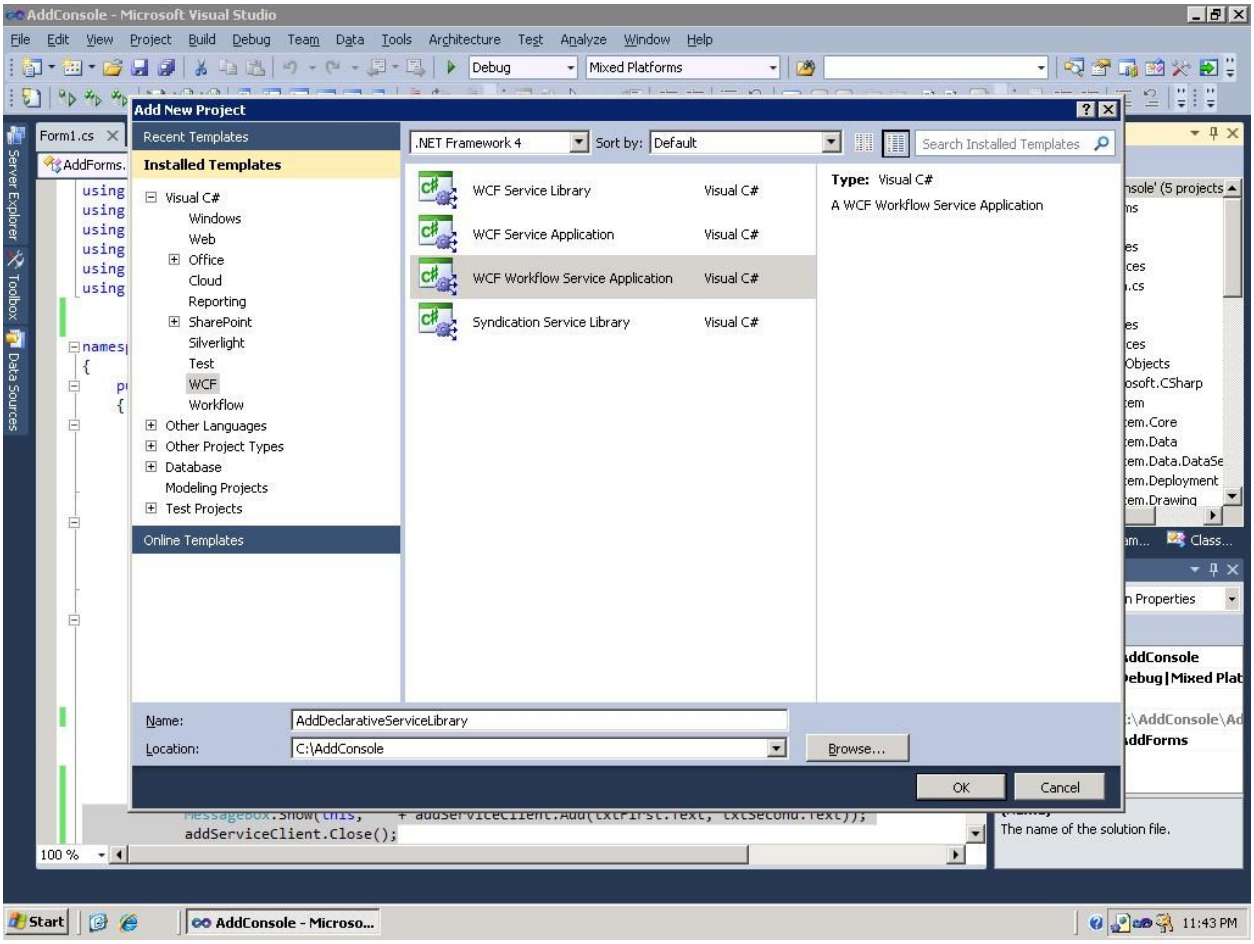
Exercitiul 1: Scrieti un cod care afiseaza doar numerele pare din intervalul 1 – 10. Pont: Folositi functia map.

Episodul al zecelea– Workflow cu WCF

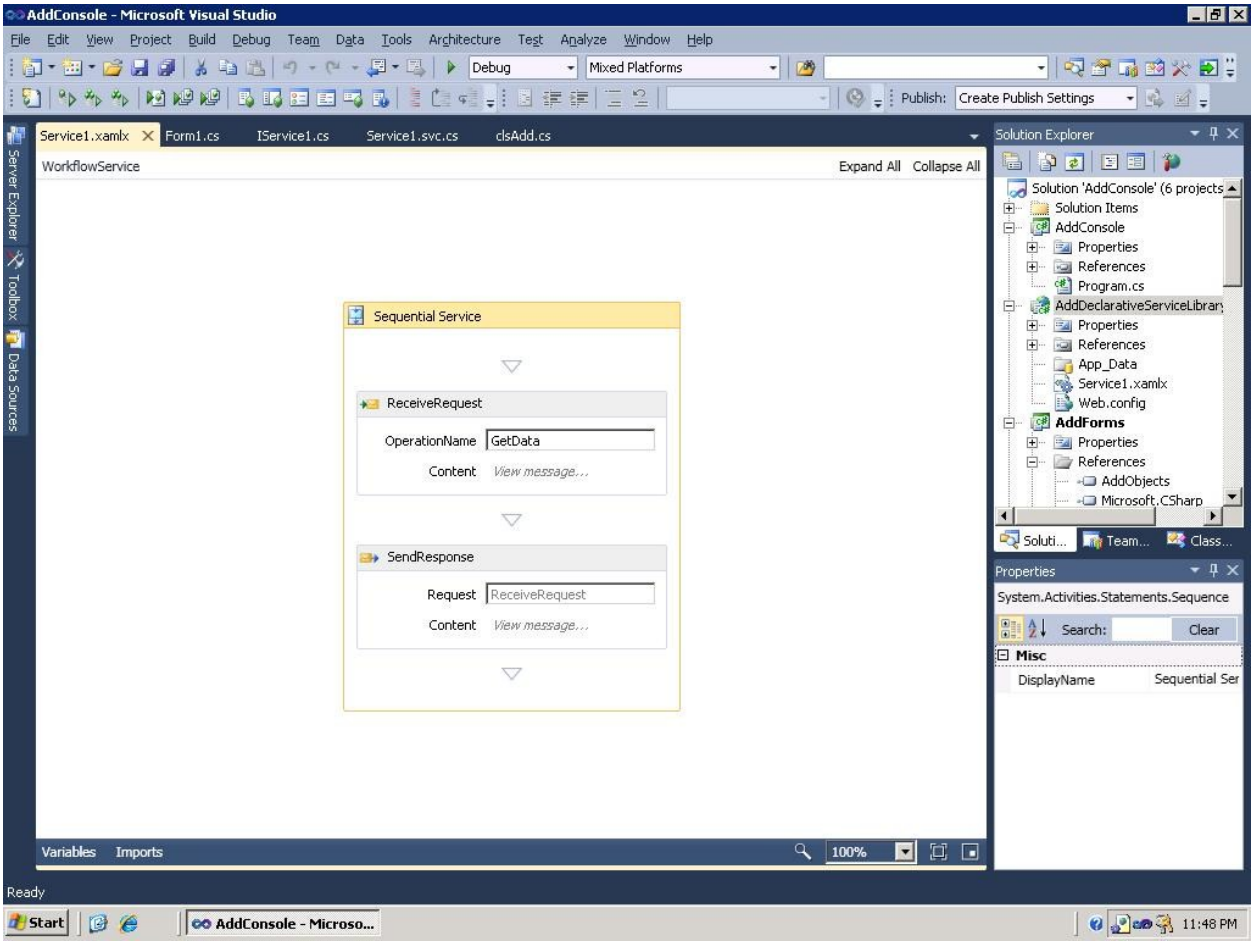
Ion are o idee, devine obositor sa mearga la Popescu pentru orice modificare la algoritmul de adunare. Nu poate gasi Popescu o modalitate de reprezentare a logicii programului care sa nu necesite compilare, si sa mearga si fara Visual Studio?

Popescu se gindeste si ii vine o idee, Windows Workflows, sint repreznтate in un fisier XML, care e compilat la rulare, pot fi editate direct cu notepad sau se poae face o aplicatie care sa hosteze editorul visual de workflows.

Si se pune pe treaba, un proiect nou din template WCF Worflow Service Application

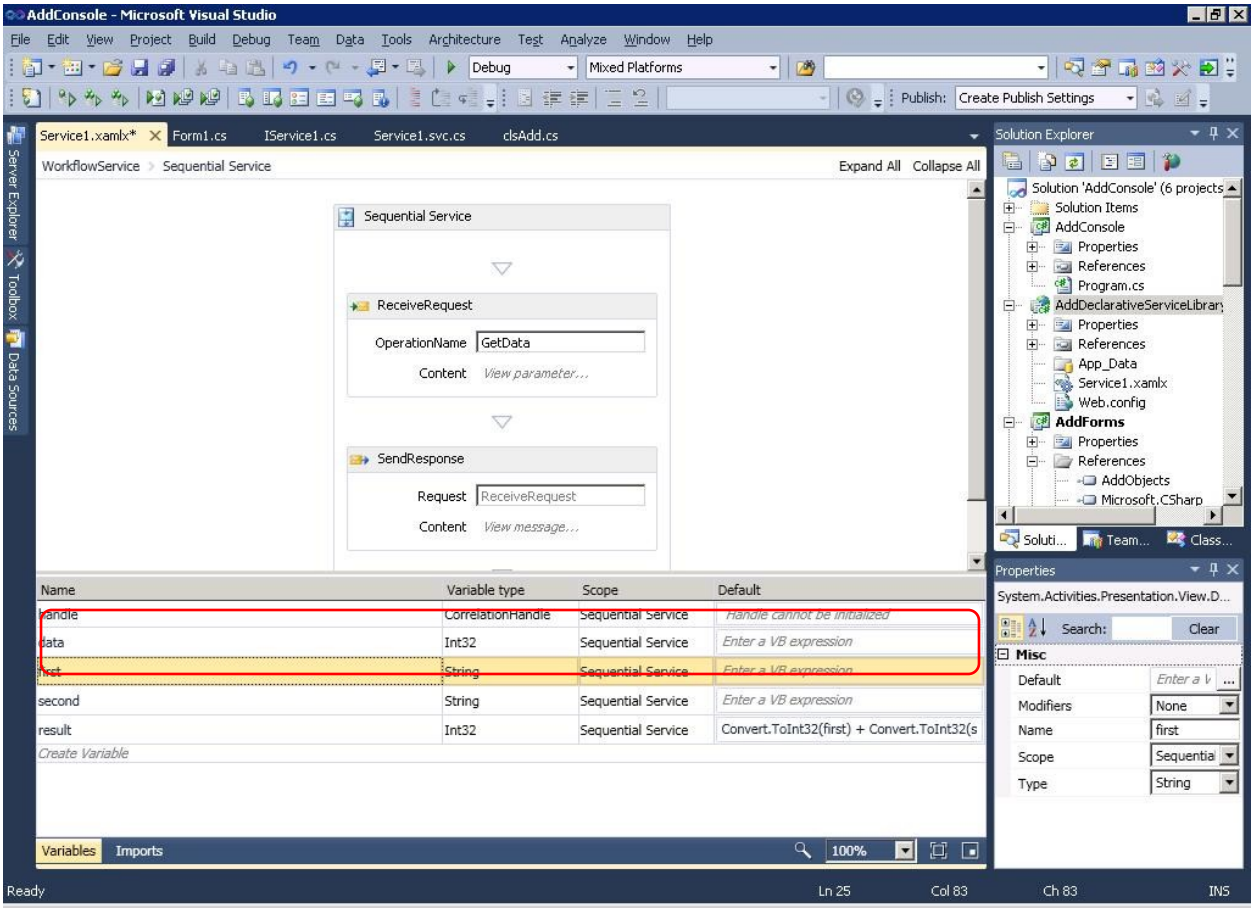


Si se trezeste cu un workflow....

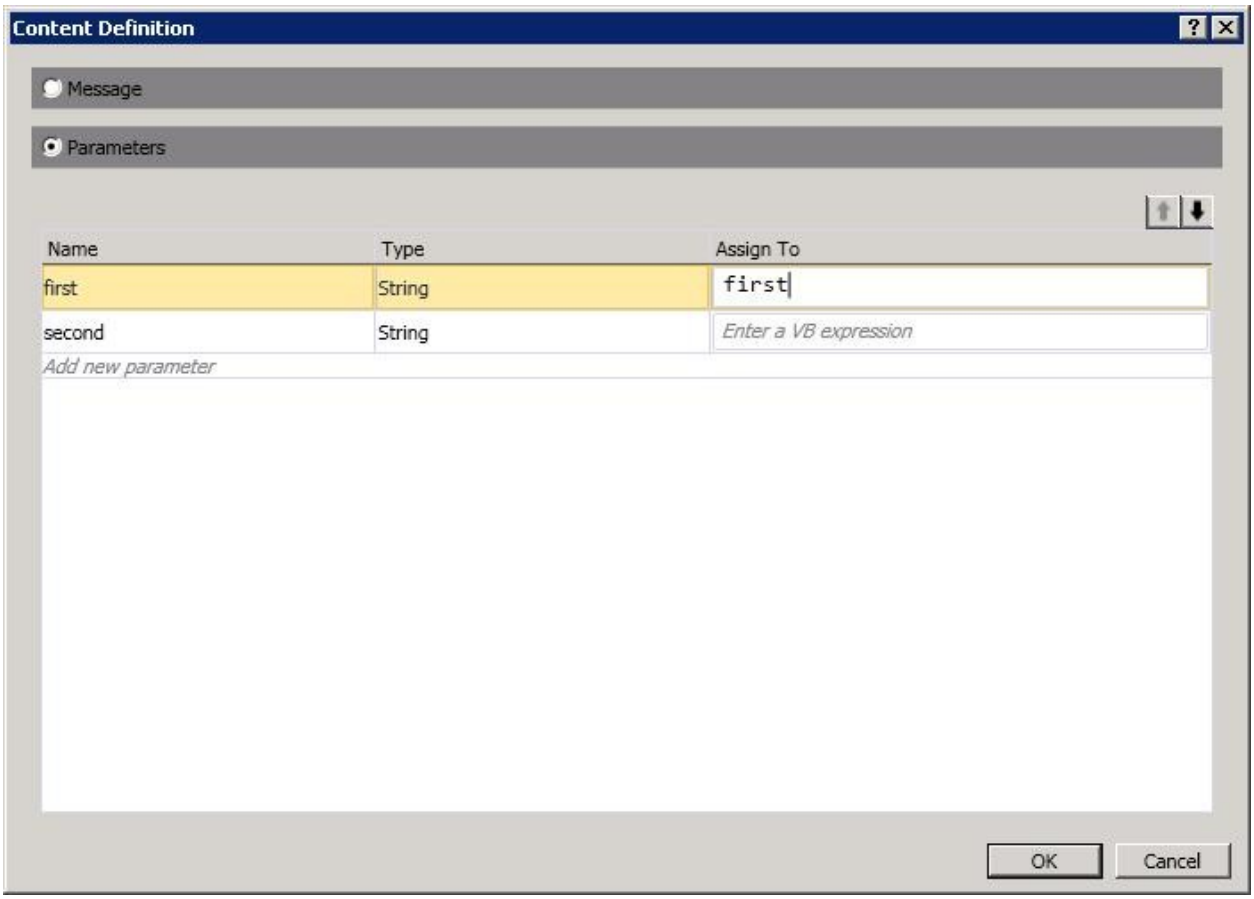


Secvential ☺. .Net 4.0 suporta worklow seventiale, flowchart si se pare ca in curind, din nou state workflows. Nota: In Visual Studio 2010 se pot folosi si workflows din 3.5.
Template de proiect folosit creaza proiect care expune un workflow secvential ca serviciu WCF.
Popesc se pune pe treaba

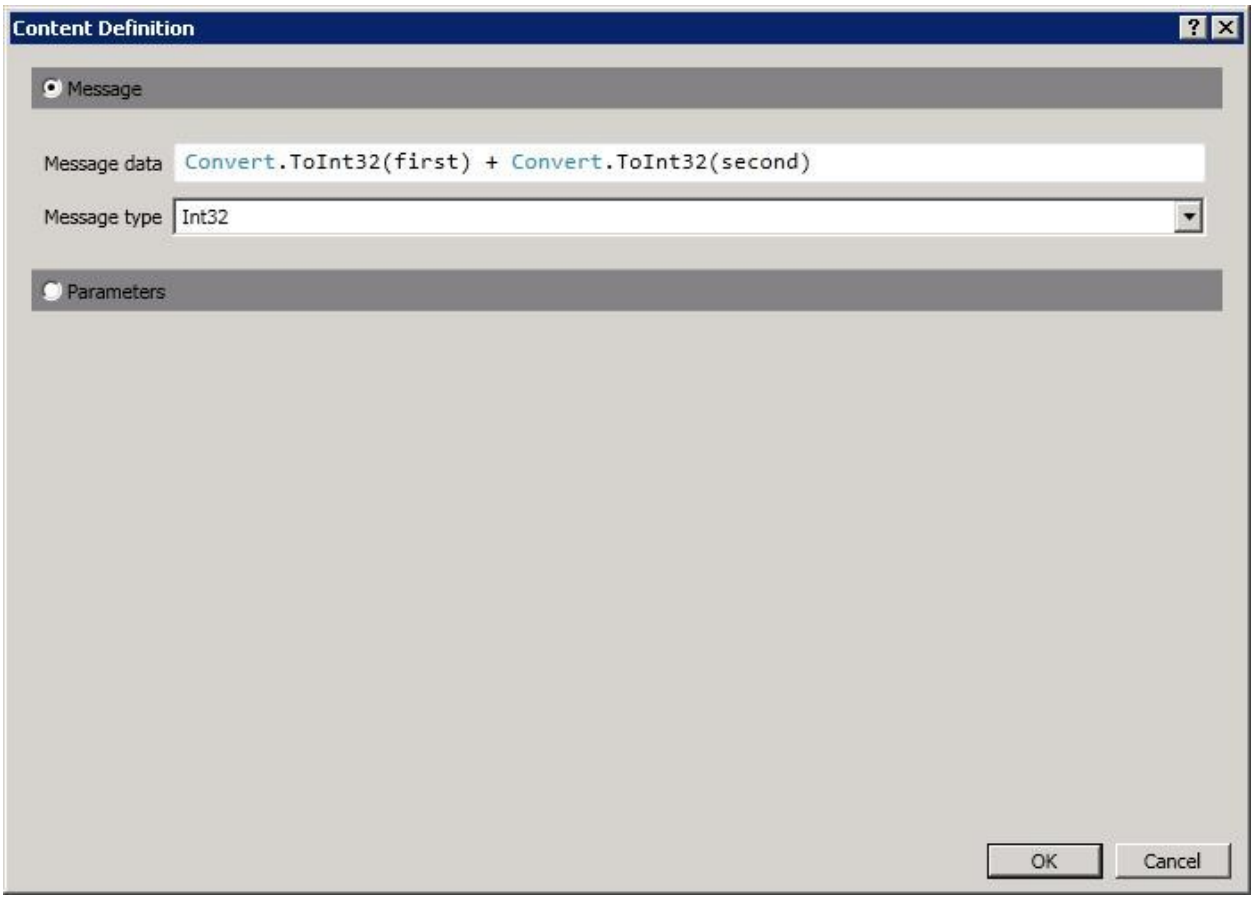
O data sa creeze varibile care sa pastreze valorile pasate ca parametrii



OK acuma la receive request, click pe View Parameter, sa creeze parametrii de intrare si sa ii lege de varibilele definite

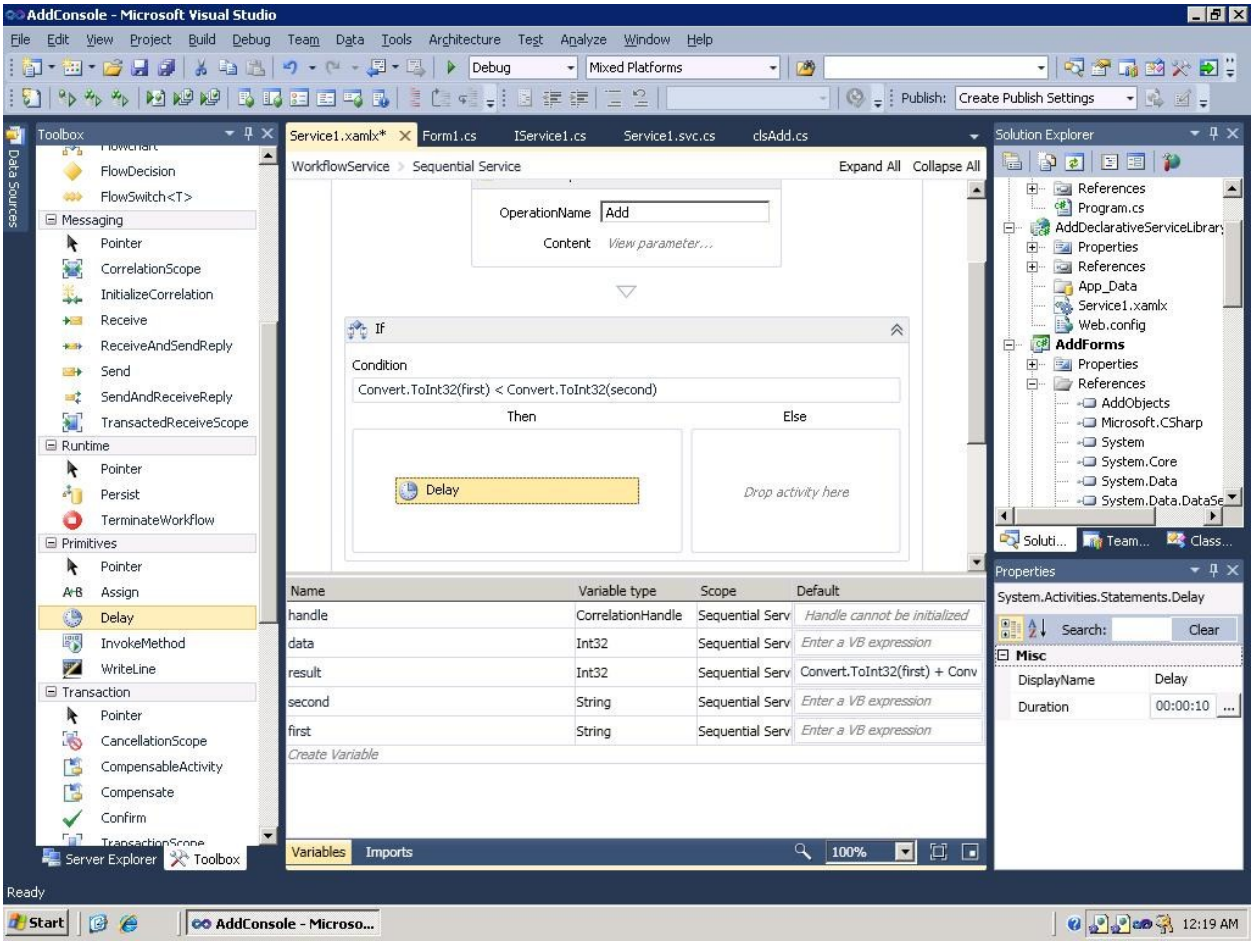


Si normal la Send Response vrem sa returneze rezultatul, click View Message si

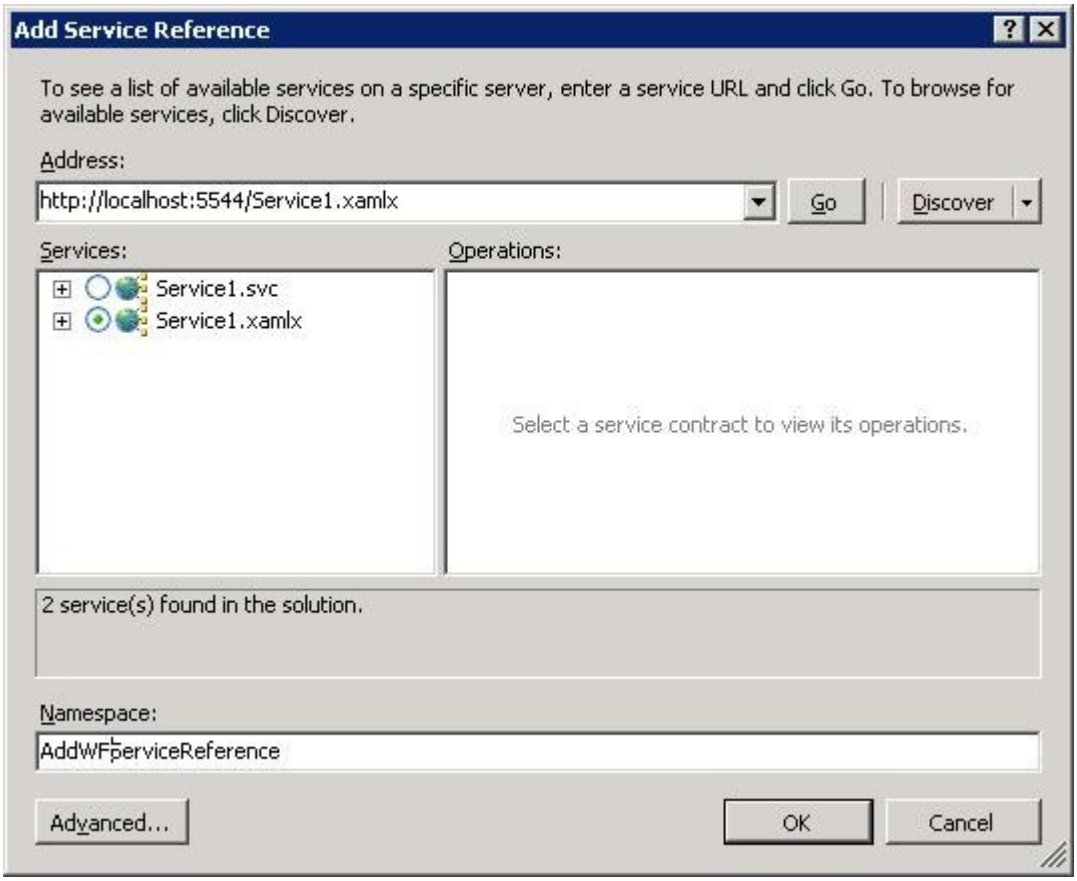


Dar sa complicam putin daca primul numar este mai mic decit cel de al doilea sa introducem un delay de 10 secunde. Deci putin drag & drop din toolbox un if si un delay. Observam ca putem avea si un flowchart workflow apelat in un worklfow secvential in .Net 4.0.

http://ronua.ro/src=babysteps



Si gata aici, sa vedem cum merge, mergem iarasi la proiectul AddForms , adaugam un service reference (exercitiu de memorie cum se face).



Si in handlerul `btnAdd_Click acum punem acest cod`

```
AddWFServiceReference.ServiceClient addServiceClient = new
AddWFServiceReference.ServiceClient();
AddWFServiceReference.Add dataToAdd = new AddWFServiceReference.Add();
dataToAdd.first = txtFirst.Text;
dataToAdd.second = txtSecond.Text;
MessageBox.Show(this, ""+addServiceClient.Add(dataToAdd));
addServiceClient.Close();
```

Incearca codul si totul merge perfect. Ion o sa fie fericit.

Exercitiu 1: Ce valori o sa aiba variabila workflow result? De ce?

Exercitiu 2: Modificati call WCF sa fie asincron. Hint: Windows Phone 7 sau Silverlight pur si simplu.

Exercitiu 3: Modificati workflow asa incit sa trimita notificari pe mail in caul in care valoare 2 mai mare egala decit prima valoare.

Episodul al unsprezecelea– Sql Server CLR

Si tocmai cand Popescu se gandea ca nu mai are ce sa faca la proiect vine iarasi Ion:

Ion: Stii, am niste Admini la baza de date SQL care ar vrea si ei sa foloseasca functia ta, in varianta aia de baza, cand aduna doua numere. Ce zici, poti sa ii ajuti ?

Popescu: Pfff. Si astia ? Sigur se rezolva...

Ion: Cam pe cand sa le zic ca e gata ?

Popescu: Pai daca nu intervine altceva, in 15 minute le termin functia plus modul de instalare/ folosire.

Ion: Bravo mai Popescule, tu ziceai ca esti incepator ?? Noooo, nu se poate. Ia sa te propun eu pentru o marire de salariu.

Popescu rase in sine lui, imediat si-a amintit ca rasfoise el odata www.infovalutar.ro si vazuse acolo ca un web service poate fi consumat intro functie CLR. Ba mai mult, functia asta CLR sa fie importata direct in serverul SQL ca si un assambly. Pai daca aia se putea face, functia lui era floare la ureche.

Primul pas a fost deschiderea VS2010 Professional si crearea unui nou proiect de tipul SQL Server Project. Doar asa se putea sigura ca dll-ul generat de catre Visual Studio va putea fi folosit direct in SQL Server. La proiectul creat adauga un New Item de tipul User-Defined Function in care adauga codul necesar.

```
[Microsoft.SqlServer.Server.SqlFunction]
    public static int Add(string first, string second)
    {
        int f = int.Parse(first);
        int s = int.Parse(second);
        return f + s;
    }
```

Dupa un build rapid in directorul aplicatiei\bin\Debug au rezultat doua fisiere: FunctiaCLRPopescu.dll si FunctiaCLRPopescu.pdb. Fisierul dll il da Popescu lui Ion impreuna cu scriptul necesar pentru folosirea lor direct in SQL. Scriptul SQL contine urmatoarele:

1. Se copiaza fisierul dll pe C:\
2. Se deschide SQL Server Management Studio si se face conectarea la instanta locala de SQL Server Express.

Nota: Adminii SQL stiu ca integrarea CLR nu este activata initial si in prealabil activeaza aceasta functionalitate

```
sp_configure 'clr enabled', 1
GO
RECONFIGURE
```

```
GO
```

3. Se adauga dll-ul primit ca si assambly pe Serverul SQL

```
Create ASSEMBLY FuntiaCLR FROM 'C:\FunctiaCLRPopescu.dll'
WITH PERMISSION_SET = EXTERNAL_ACCESS
```

Nota: In caz ca se doreste readaugarea dll-ului (versiune noua) stergerea assambly-ului se face astfel

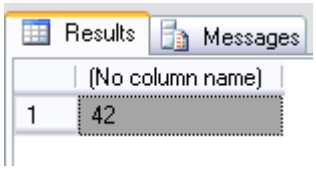
```
    IF EXISTS (SELECT name FROM sys.assemblies WHERE name = 'FuntiaCLR')
    DROP ASSEMBLY FuntiaCLR
GO
```

4. Se creeaza o funtie SQL care sa foloseasca assambly-ul

```
CREATE FUNCTION AddSQL(@f NVARCHAR(10),
                       @s NVARCHAR(10))
RETURNS INT
AS EXTERNAL NAME FuntiaCLR.UserDefinedFunctions.[Add]
```

5. Testingul

```
SELECT dbo.AddSQL(11,31)
```



Popescu rasufla usurat. Inca un job finalizat, inca un pas spre a deveni un meserias. Cine stie ce ii mai rezerva ziua de maine.

Exercitiu 1: ce-ar fi sa il ajutati pe Popescu sa modifice functia sa faca si diferenta ? Sigur vin astia cu cerinta maine.

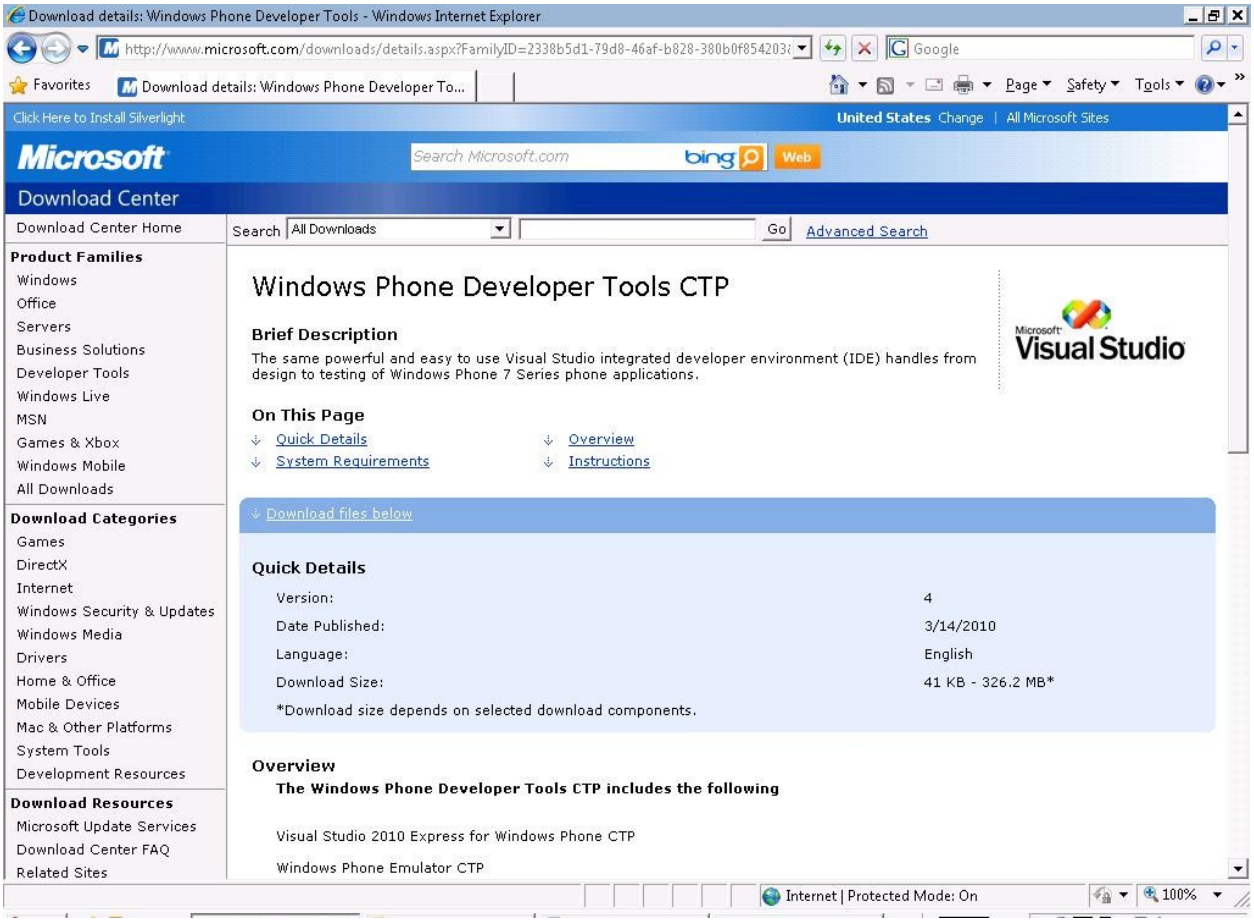
Episodul al doisprezecelea – Window Phone 7

Partea intii

La Popescu vine ... Da ati ghicit, iarasi Ion, cel de la vinzari, cu o constatare. Toata lumea are un smartphone, si Popescule noi ce ne facem?
Popescu se pune pe treaba si afla ca, cu Visual Studio Express 2010 poate dezvolta aplicatii pentru Windows Phone 7.
Dar de unde sa inceapa ? Pai a gasit site pentru developeri de Windows Phone (http://developer.windowsphone.com/).
Si dupa o privire constata ca trebuie sa downloadeze Windows Phone Developer Tools CTP

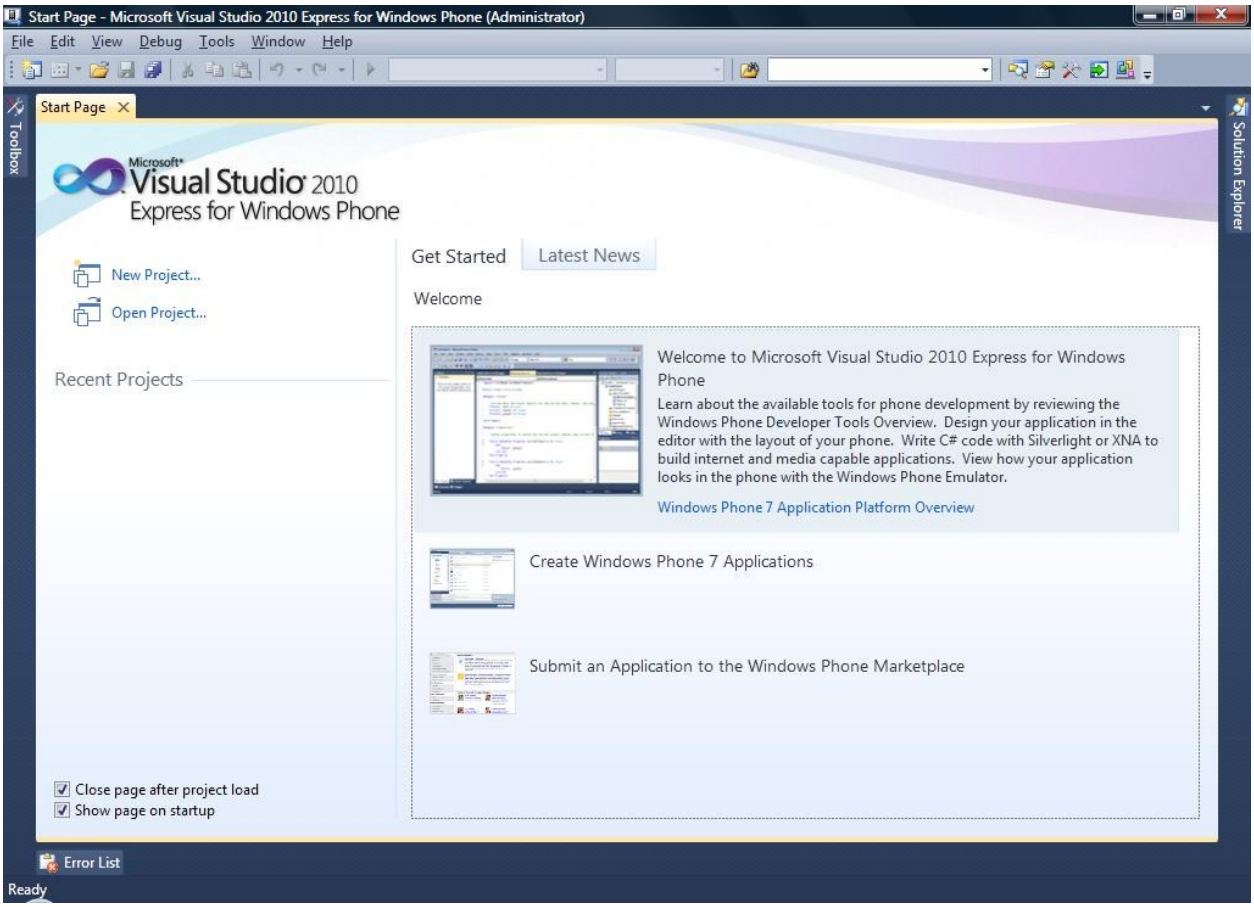
http://ronua.ro/src=babysteps

(http://www.microsoft.com/downloads/details.aspx?FamilyID=2338b5d1-79d8-46af-b828-380b0f854203&displaylang=en). Nota: se va putea dezvolta pentru Windows Phone 7 cu versiunile Express de Visual, spre deosebire de dezvoltarea vizind Windows Mobile de pina acuma care necesita versiuni Professional sau mai bune.
Atentie la system requirements, emulatorul este destul de pretentios: Vista sau Window 7, 2 GB Ram, DirectX 10 suportat de placa video (rulati DxDiag, tab Display, DDI Version trebuie sa fie 10 sau mai buna).
Si deci dupa ce Popescu a capatat un calculator nou s-a apucat de treaba.
Popescu descarca Windows Phone Developer
Tools:

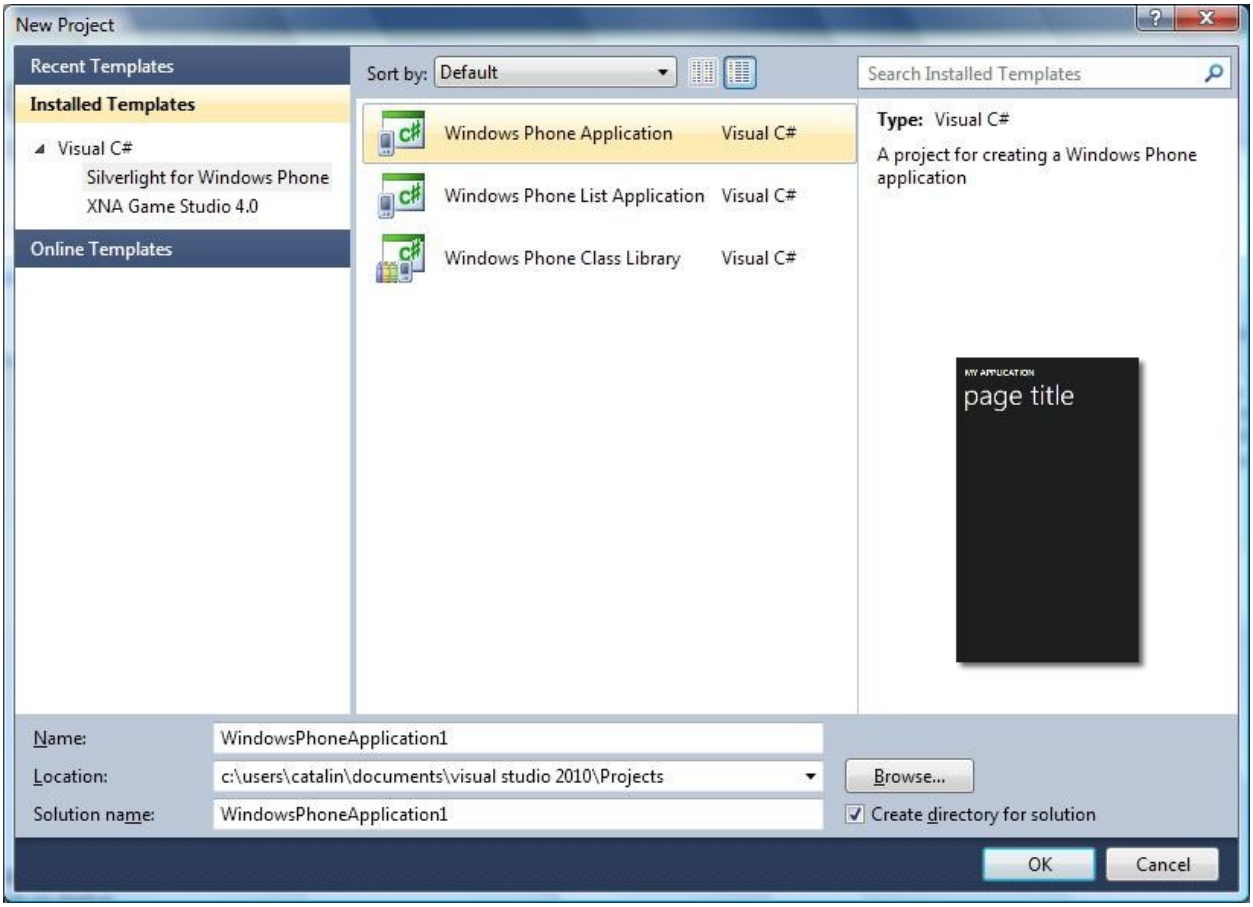


http://ronua.ro/src=babysteps

A pornit Visual Studio 2010 Express (for Windows Phone):

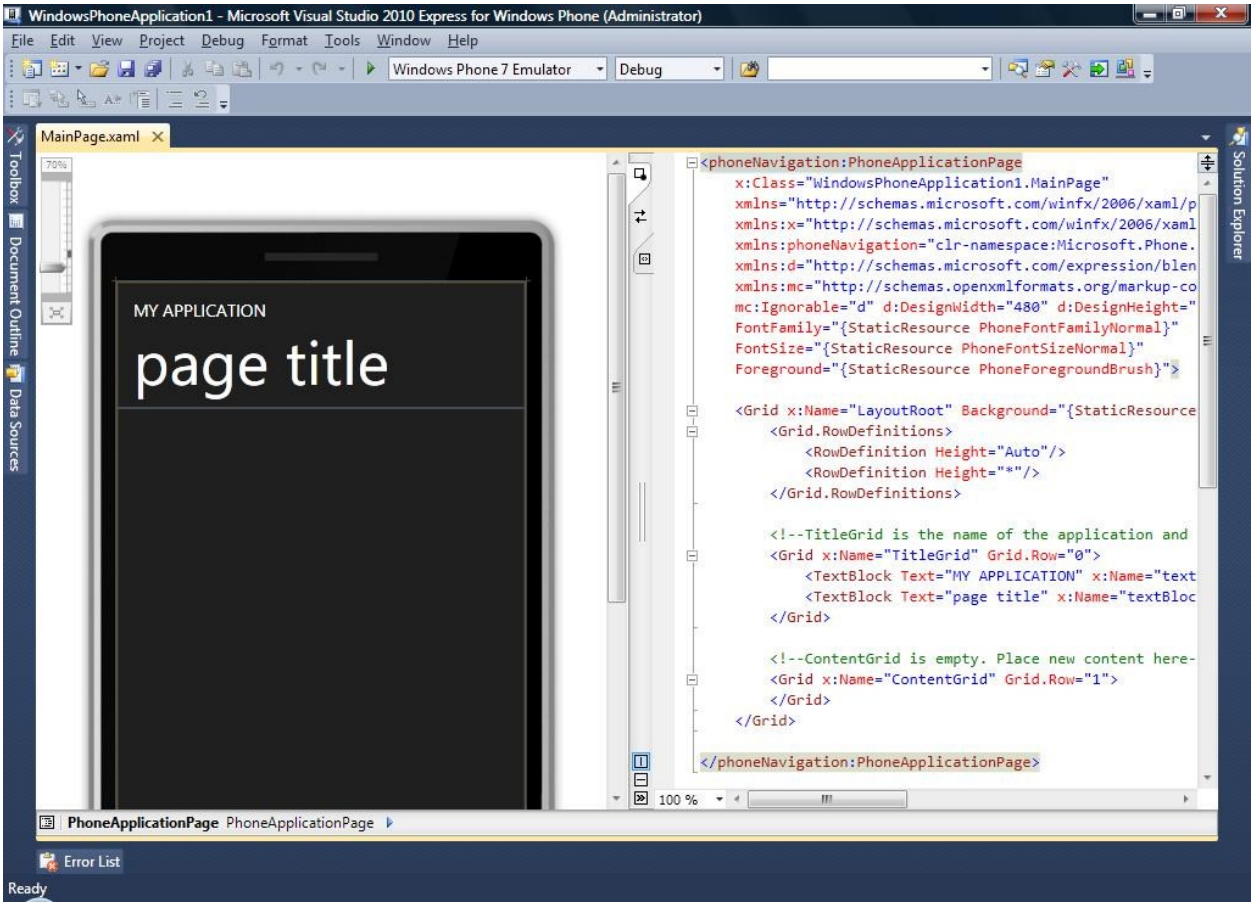


Si acuma Popescu se apuca sa creeze un noi proiect, si se decide ca Silverlight, e mai aproape de ce stie el acuma.

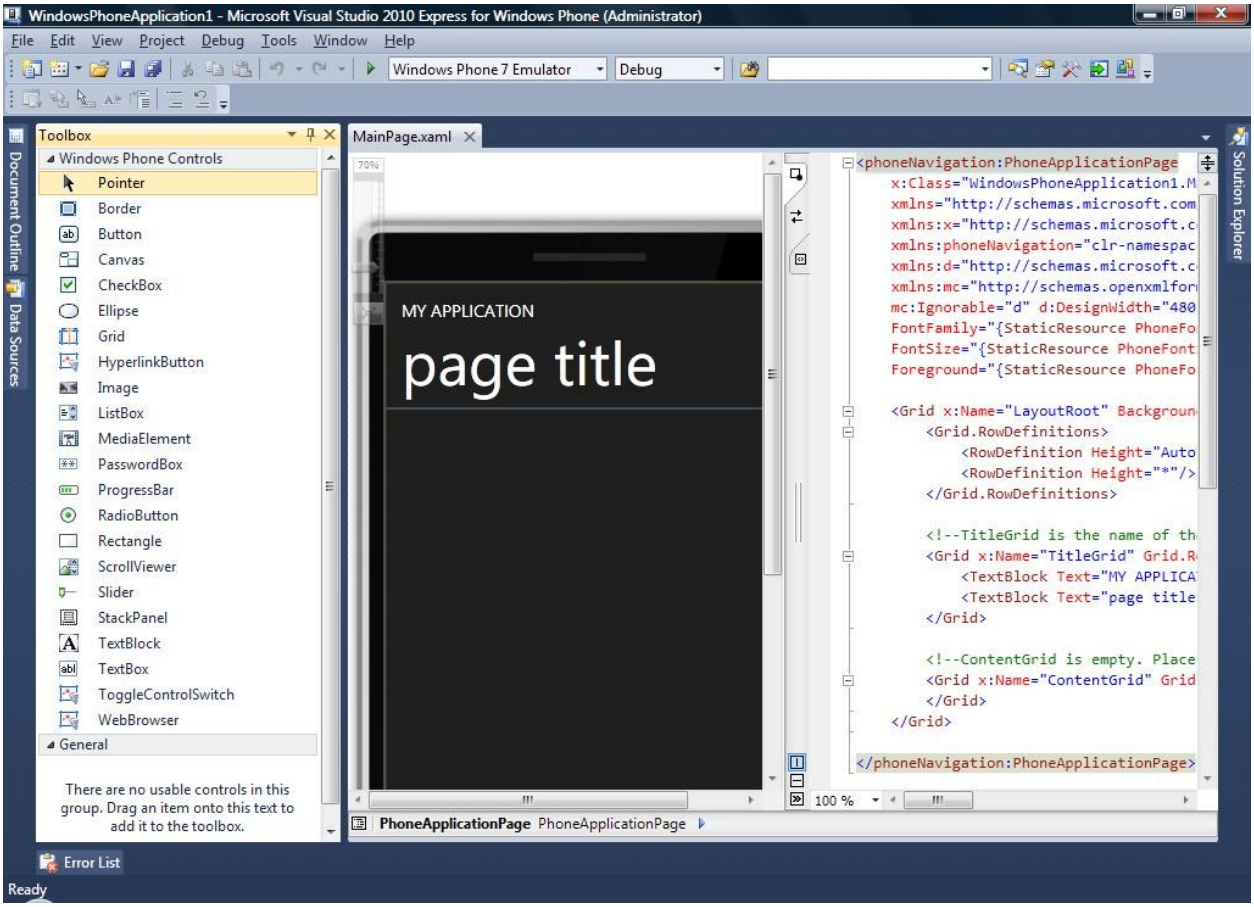


Care arata necustomizat

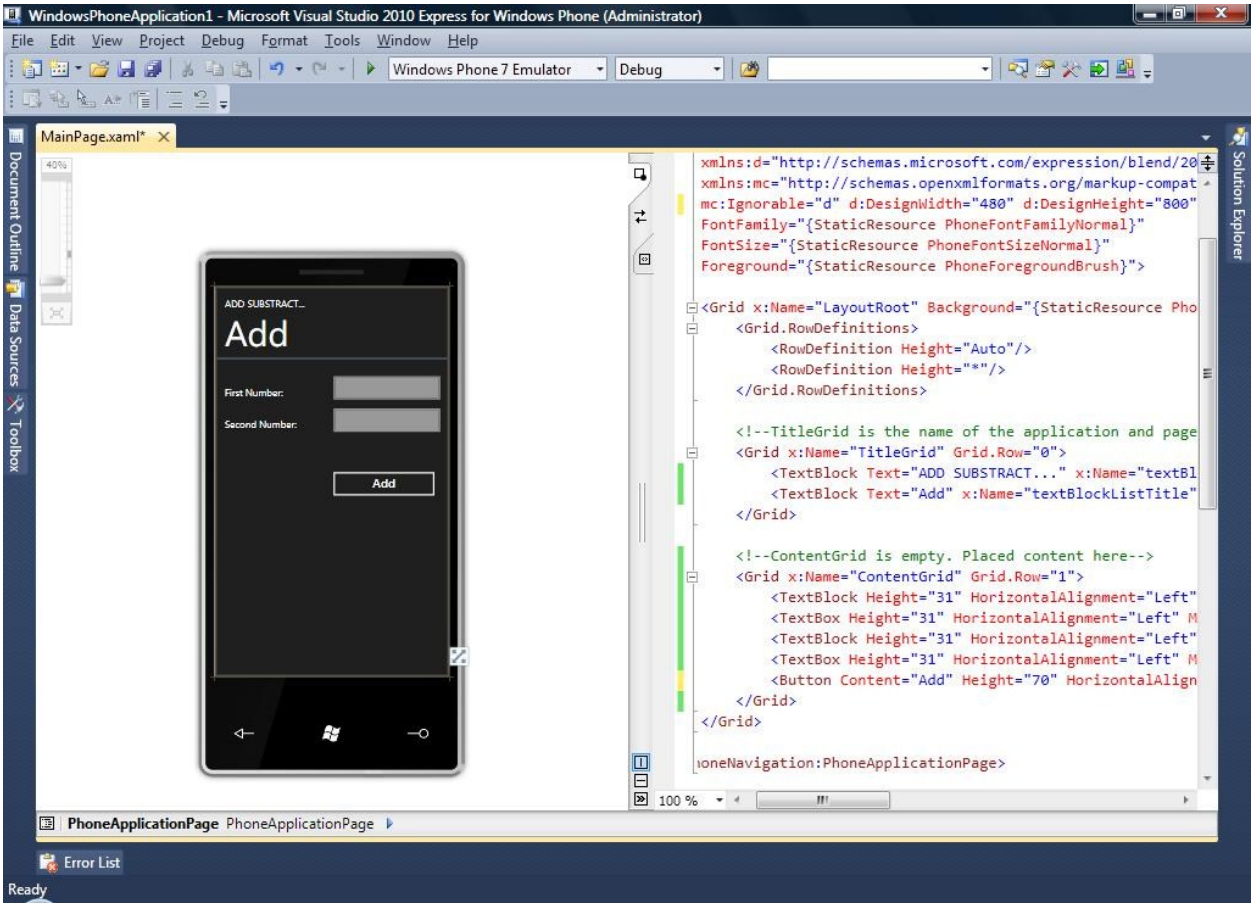
asa



Sa vedem ce controale avem la dispozitie in Windows Phone



Popescu presupune ca Label din Forms e TextBlock, si restul ii pare cunoscut si se pune pe treaba, drag & drop.



Si descopera ca e foarte usor acuma sa modifice proprietatile in editorul de xaml din dreapta. A facut si double click pe butonul adaugat si numit Add ca sa creeze handlerul de event.

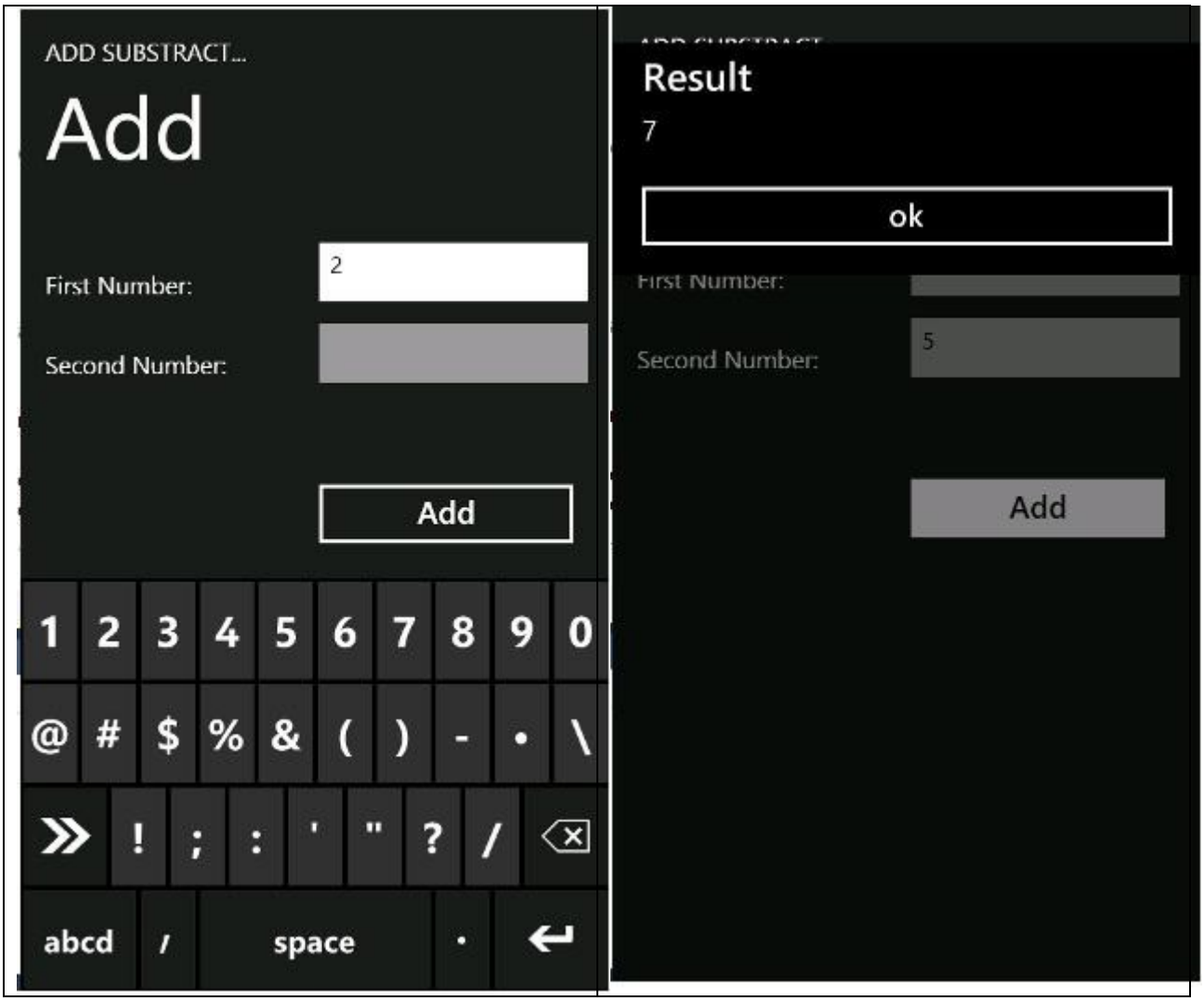
```
<!--TitleGrid is the name of the application and page title-->
        <Grid x:Name="TitleGrid" Grid.Row="0">
            <TextBlock Text="ADD SUBSTRACT..." x:Name="textBlockPageTitle"
Style="{StaticResource PhoneTextPageTitle1Style}"/>
            <TextBlock Text="Add" x:Name="textBlockListTitle"
Style="{StaticResource PhoneTextPageTitle2Style}"/>
        </Grid>

        <!--ContentGrid. Placed content here-->
        <Grid x:Name="ContentGrid" Grid.Row="1">
            <TextBlock Height="31" HorizontalAlignment="Left"
Margin="20,56,0,0" Name="textBlockNumber1" Text="First Number:"
VerticalAlignment="Top" />
            <TextBox Height="31" HorizontalAlignment="Left"
Margin="231,24,0,0" Name="textBoxNumber1" Text="" VerticalAlignment="Top"
Width="243" >
                <TextBox.InputScope>
                    <InputScope>
                        <InputScope.Names>
                            <InputScopeName NameValue="Number"/>
                        </InputScope.Names>
```

```
                    </InputScope>
                </TextBox.InputScope>
            </TextBox>
            <TextBlock Height="31" HorizontalAlignment="Left"
Margin="20,121,0,0" Name="textBlockNumber2" Text="Second Number:"
VerticalAlignment="Top" />
            <TextBox Height="31" HorizontalAlignment="Left"
Margin="231,90,0,0" Name="textBoxNumber2" Text="" VerticalAlignment="Top"
Width="243" >
                <TextBox.InputScope>
                    <InputScope>
                        <InputScope.Names>
                            <InputScopeName NameValue="Number"/>
                        </InputScope.Names>
                    </InputScope>
                </TextBox.InputScope>
            </TextBox>
            <Button Content="Add" Height="70" HorizontalAlignment="Left"
Margin="231,221,0,0" Name="buttonAdd" VerticalAlignment="Top" Width="231"
Click="buttonAdd_Click" />
        </Grid>
```

Si cum pomeneam de handlerul de event click, acuma ar trebui sa scriem si un cod care face operatiile. Class library (dll) creat anterior nu poate fi folosit asa cum este, nu avem compatibilitate binara ci doar al nivel de cod sursa si si asta pina la un punct (.Net Compact nu implmenteaza toate clasele si metodele din .Net). Si pentru ca in acest caz nu o sa avem o aplicatie consola, deocamdata scriem direct codul.

```
private void buttonAdd_Click(object sender, RoutedEventArgs e)
        {
       int f = int.Parse(textBoxNumber1.Text,
System.Globalization.NumberStyles.AllowThousands);
             int s = int.Parse(textBoxNumber1.Text,
System.Globalization.NumberStyles.AllowThousands);
              MessageBox.Show((f + s).ToString(),"Result",MessageBoxButton.OK);
        }
```

Singura mare diferenta e ca nu mai e folosita cultura, CurrentThread nu mai exista in Windows Phone. Dar si-a dat seama ca asta cu cultura e o discutie lunga si cum aplicatia merge, este satisfacut ca la lansarea Windows Phone 7 vor fi pe val.



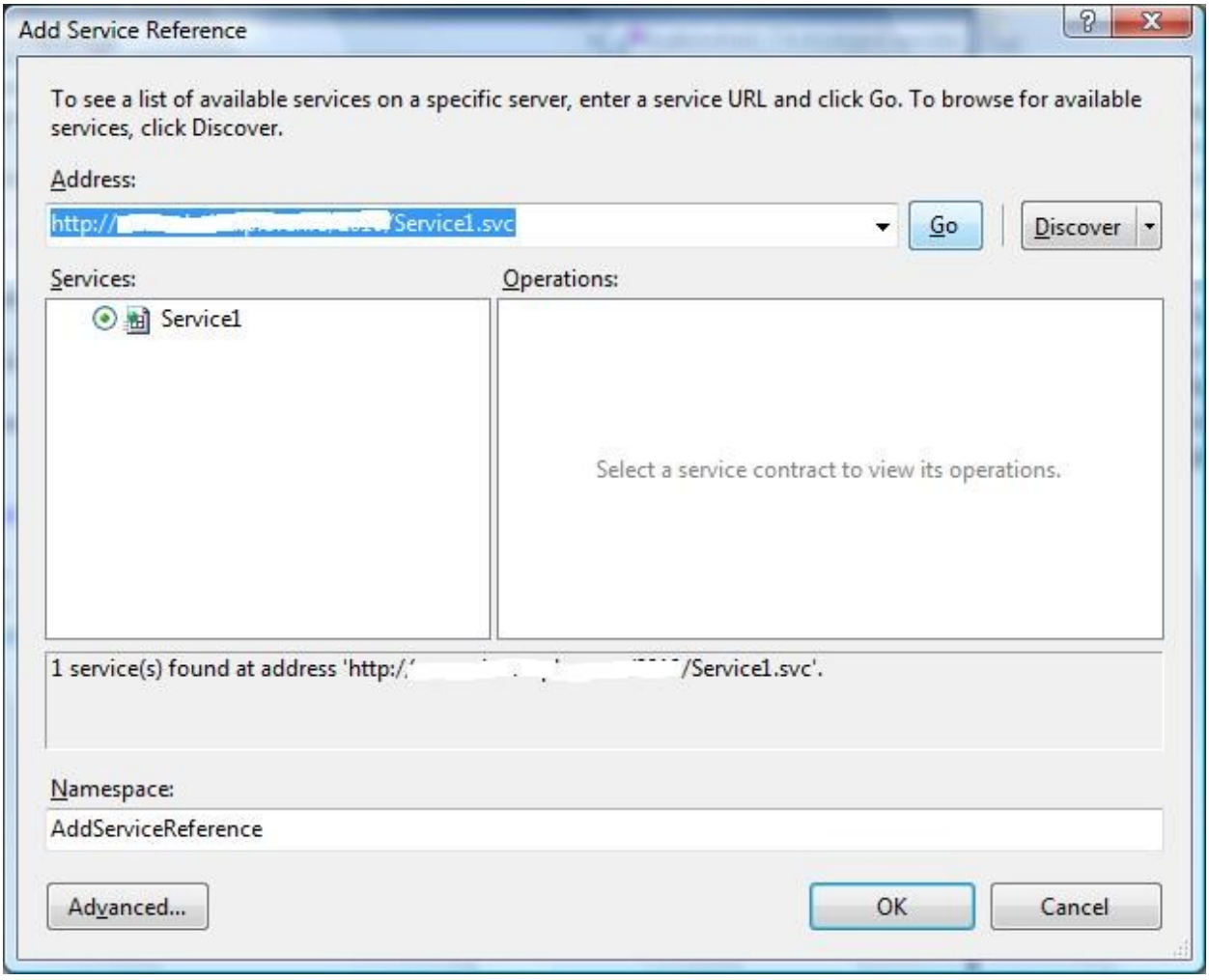
Exercitiu 1: Creati clasa clsAdd pentru noua platforma si folositi-o in aplicatie.

Exercitiu 2: Studiati adaugarea aplicatiilor la Windows Marketplace for Mobile (https://marketplace.windowsphone.com/Default.aspx)

Exercitiu 3: Scrieti o aplicatie desktop WPF (sau Silverlight desktop) care executa aceeasi operatii, hint se foloseste alt template de proiect dar codul e la fel aproape, cum ar zice unii leveraging knowledge merge in ambele directii ☺.

Partea a doua : Windows Phone 7 consumind un WCF Service

La Popescu continua cu modificare aplicatiilor sa foloseasca serviciul WC si ajunge la Windows Phone 7. Si primul pas e sa adauge o referenta la serviciu.



Si citeste putin pe blogurile de pe user group lui preferat si afla care sint diferentele la consumul unsui serviciu WCF in o aplicatie Silverlight Windows Phone .

1. Nu are nevoie de un fisier de „cross domain policy” pe serverul care hosteaza serviciul WCF, spre deosebire de aplicatiile Silverlight „normale”.
2. Ca la toate aplicatiile Silverlight, este permis doar apelul asincron.

Si asa arata codul in acest caz:

```
private void buttonAdd_Click(object sender, RoutedEventArgs e)
        {
            AddServiceReference.Service1Client client = new
AddServiceReference.Service1Client();
```

```
            client.AddNumbersCompleted += new
EventHandler<AddServiceReference.AddNumbersCompletedEventArgs>(AddNumbersCompl
eted);

            client.AddNumbersAsync(textBoxNumber1.Text, textBoxNumber2.Text);
        }

        void AddNumbersCompleted(object sender,
AddServiceReference.AddNumbersCompletedEventArgs e)
        {
            MessageBox.Show(e.Result.ToString());
        }
```

Ca modificare importanta AddNumbersCompleted care se executa cind serviciul WCF isi termina apelul.

Episodul al treisprezecelea – WPF

Aici Popescu si-a dat singur de lucru – auzise de WPF, citise – si si-a zis ca aplicatia aceasta e numai buna de portat peste WPF.

Popescu a citit despre sablonul M-V-VM si stiu ca la baza sta modelul, datele de intrare. Ca atare crea o clasa pentru aceasta :

```
    public class Addition
    {
        public int? Operand1 { get; set; }
        public int? Operand2 { get; set; }
    }
```

Baiat destept, Popescu a realizat ca initial datele pot lipsi si a ales un tip nulabil de date, System.Nullable<int> sau varianta mai succinta : „int?”.

Dupa aceasta el lua urmatoarea litera la rand, V-ul, si crea un view potrivit :

```
<Window x:Class="WPF_Addition_Simple.Views.InitialMain"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
```

```
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="Auto"/>
        </Grid.ColumnDefinitions>

        <TextBlock Grid.Row="0" Grid.Column="0">Operand 1 :</TextBlock>
        <TextBlock Grid.Row="1" Grid.Column="0">Operand 2 :</TextBlock>
        <Button Grid.Row="2" Grid.Column="0"
Margin="0,0,10,0">Compute</Button>

        <TextBox Grid.Row="0" Grid.Column="1" Width="100" Text="{Binding
Operand1}"/>
        <TextBox Grid.Row="1" Grid.Column="1" Width="100" Text="{Binding
Operand2}"/>
        <TextBlock Grid.Row="2" Grid.Column="1" Text="{Binding Result}"/>

    </Grid>
</Window>
```

A trecut usor peste confuzia ca Grid ar fi un DataGrid si a inteles ca este un element destinat pozitionarii altor elemente, un fel de Table din HTML.

A invatat despre databinding-ul mai puternic si care permite o sintaxa declarativa, introdus in WPF si a facut uz de el pentru a lipi textul pentru caseta cu operanzi si pentru eticheta cu rezultatul.

Se gandi el o vreme cum sa faca cu calculul efectiv al operatiei si prima tentatie a fost sa dea dublu clic in designer pe butonul “Compute” insa a realizat ca code behind-ul este incompatibil cu sablonul MVVM. Asa ca se gandi ca va crea o comanda potrivita pentru acel buton. Asadar a actualizat markup-ul corespunzator :

```
        <Button Grid.Row="2" Grid.Column="0" Margin="0,0,10,0"
Command="{Binding Compute}">Compute</Button>
```

Acum Popescu se apuca de munca cea mai grea in WPF : dezvoltarea ultimelor doua litere, VM, a **v**iew**m**odel-ului. A realizat ca va trebui realizata o comanda (ICommand) care sa ii faca calculul efectiv (citise despre comenzi inainte). Scrise o prima varianta :

```
    public class AdditionViewModel
    {
        private readonly Addition model;
        public ICommand Compute { get; private set; }
        public int? Result { get; private set; }
```

```
        public AdditionViewModel(Addition model)
        {
            this.model = model;
        }

        public int? Operand1
        {
            get { return model.Operand1; }
            set { model.Operand1 = value; }
        }

        public int? Operand2
        {
            get { return model.Operand2; }
            set { model.Operand2 = value; }
        }
    }
```

Stia Popescu ca viewmodel-ul nu are voie sa expuna efectiv instanta de model ci doar acele proprietati ale lui, relevante view-ului cu care viewmodel-ul face pereche. Se gandi el o vreme cum sa faca cu comanda si apucase sa vada pe net niste implementari potrivite pentru interfata ICommand :

```
    public class DelegateCommand : ICommand
    {
        private readonly Action execute;
        private readonly Predicate<object> canExecute;

        public DelegateCommand(Action execute) : this(execute, null) { }

        public DelegateCommand(Action execute, Predicate<object> canExecute)
        {
            if (execute == null)
            {
                throw new ArgumentNullException("execute");
            }

            this.execute = execute;
            this.canExecute = canExecute;
        }

        public bool CanExecute(object parameter)
        {
            return canExecute == null ? true : canExecute(parameter);
        }

        public event EventHandler CanExecuteChanged
        {
            add { CommandManager.RequerySuggested += value; }
            remove { CommandManager.RequerySuggested -= value; }
        }

        public void Execute(object unused)
```

```
        {
            execute();
        }
    }
```

Comenzile astea in sine sunt ca niste metode insa in afara de metoda efectiva care face lucru mai pot spune UI-ului cand nu sunt folosibile (CanExecute) si cand s-a schimbat aceasta stare (CanExecuteChanged). Astfel, butoane sau alte elemente de UI ce ar putea actiona o comanda pot sa se activeze si sa se dezactiveze singure, in mod automat!

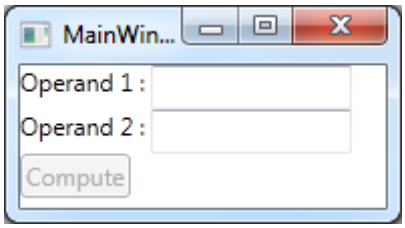
Cu implementarea DelegateCommand el poate crea o instanta de comanda potrivita pentru calculul adunarii dorite de el :

```
    public class AdditionViewModel
    {
        private readonly Addition model;
       public ICommand Compute { get; private set; }
        public int? Result { get; private set; }

        public AdditionViewModel(Addition model)
        {
            this.model = model;
            Compute = new DelegateCommand(OnCompute, CanCompute);
        }

        public int? Operand1
        {
            get { return model.Operand1; }
            set { model.Operand1 = value; }
        }

        public int? Operand2
        {
            get { return model.Operand2; }
            set { model.Operand2 = value; }
        }

        public bool CanCompute(object unused)
        {
            return model.Operand1.HasValue && model.Operand2.HasValue;
        }

        public void OnCompute()
        {
            Result = model.Operand1.Value + model.Operand2.Value;
        }
    }
```

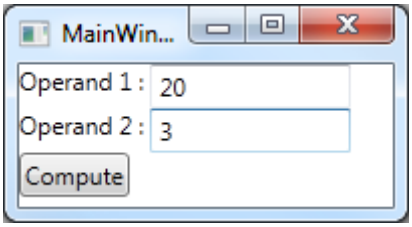
In sfarsit, trebuie sa porneasca cumva aplicatia si sa se asambleze model-ul, viewmodel-ul si view-ul. Deschise App.xaml.cs si scrise :

```
public partial class App
{
    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);
        var main = new MainWindow
        {
            DataContext = new AdditionViewModel(new Addition())
        };
        main.Show();
    }
}
```

Rula el bucuros aplicatia. A vazut ca la pornire nu e completat nici un numar si ca butonul de calcul e dezactivat. Crescu inima in el de bucurie.



Completa primul numar, il completa pe al doilea. A fost usor jenat ca a trebuit sa mute focus-ul de pe al doilea textbox ca sa se activeze butonul de calcul insa se dezumfla cand vazu ca apasand pe buton nu se intampla nimic.



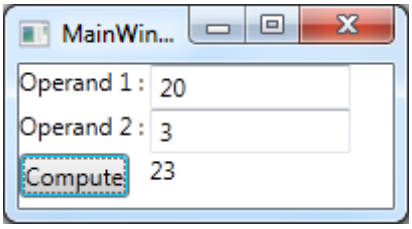
Se apuca de depanat (debug), vazu ca se executa codul bun, rezultatul e corect si e depus in membrul „Result” dar pe ecran nu aparea nimic.

Se sfatui cu Andrei (un prieten) si acesta i-a spus ca viewmodel-ul trebuie sa ii spuna cumva view-ului ca el (viewmodel-ul) si-a schimbat starea. Si sa nu se grabeasca sa scrie cod explicit pentru asta ci doar sa

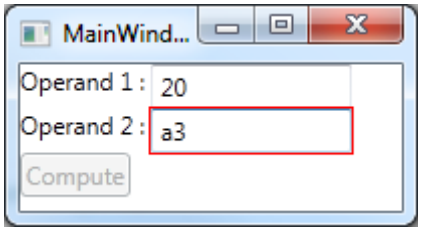
implementeze interfata INotifyPropertyChanged pe viewmodel. Circumspect Popescu se apuca de treaba :

```
public class AdditionViewModel : INotifyPropertyChanged
{
    private readonly Addition model;
    public ICommand Compute { get; private set; }
    public event PropertyChangedEventHandler PropertyChanged;
    public int? Result { get; private set; }

    public AdditionViewModel(Addition model)
    {
        this.model = model;
        Compute = new DelegateCommand(OnCompute, CanCompute);
    }

    public int? Operand1
    {
        get { return model.Operand1; }
        set { model.Operand1 = value; }
    }

    public int? Operand2
    {
        get { return model.Operand2; }
        set { model.Operand2 = value; }
    }

    public bool CanCompute(object unused)
    {
        return model.Operand1.HasValue && model.Operand2.HasValue;
    }

    public void OnCompute()
    {
        Result = model.Operand1.Value + model.Operand2.Value;
        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs("Result"));
    }
}
```

Intelese el repede ca interfata, desi expune doar un eveniment, PropertyChanged, el poate sa notifice abonatii (doar daca exista, Andrei l-a sfatuit sa faca o testare de null inainte) ca s-a schimbat valoarea unei proprietati si sa spuna si care proprietate. Rula el aplicatia din nou, cu un usor dubiu insa vazu acum ca ii este afisat rezultatul!



Era tare fericit si simtea ca a terminat de lucru si aici. Verifica din nou programul, incerca sa introduca date eronate (litere in loc de cifre) si vazu ca textbox-ul se colora in rosu.



Totusi il nemultumea faptul ca trebuia sa mute focus-ul de pe textbox ca sa se produca actualizarea si vorbi cu Andrei daca exista vreo solutie pentru asta. Andrei ii spuse sa incerce sa specifice pe binding-urile aferente PropertyChanged pe UpdateSourceTrigger :

```
        <TextBox Grid.Row="0" Grid.Column="1" Width="100" Text="{Binding
Operand1, UpdateSourceTrigger=PropertyChanged}"/>
        <TextBox Grid.Row="1" Grid.Column="1" Width="100" Text="{Binding
Operand2, UpdateSourceTrigger=PropertyChanged}"/>
```

Rula din nou aplicatia si acum, in sfarsit facea totul asa cum spera. Imediat cum completa al doilea numar, butonul de calcul devenea folosibil.

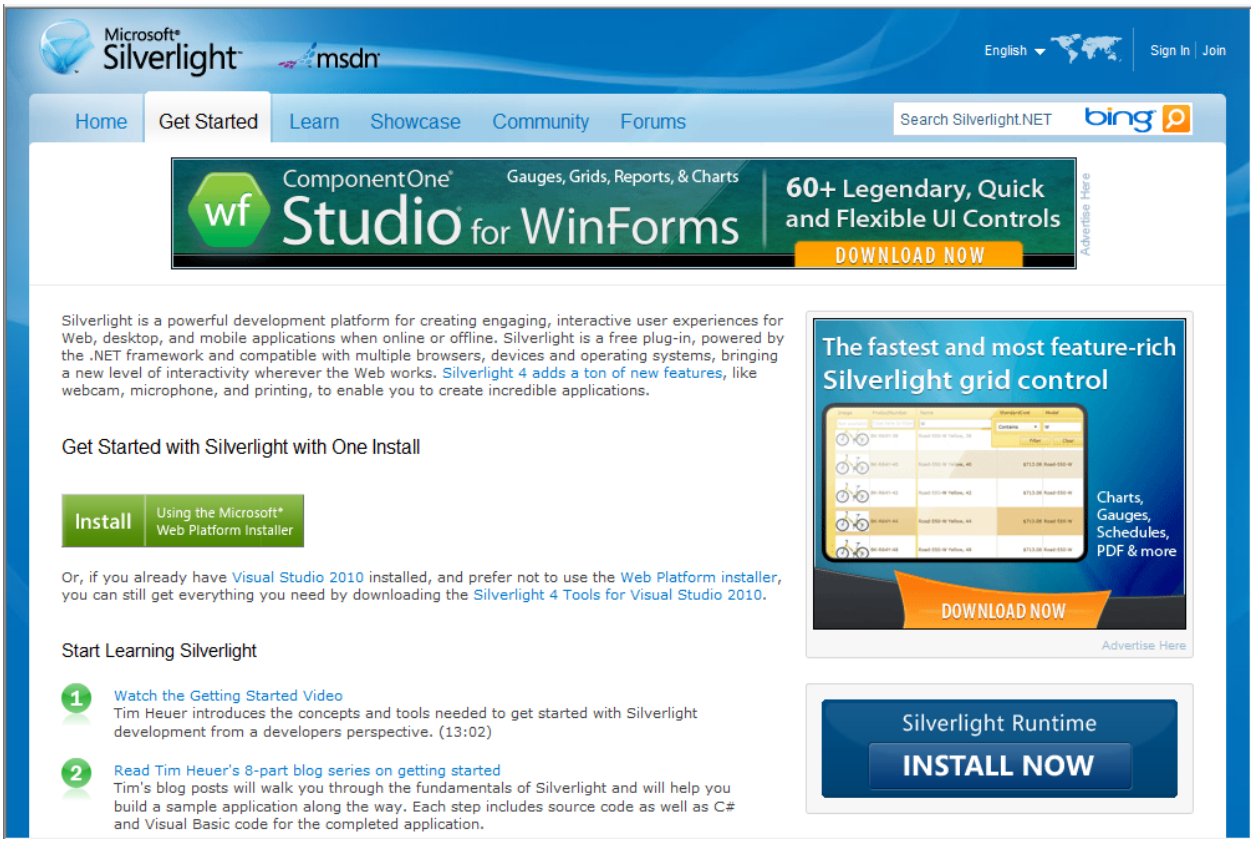
Totul fiind rezolvat, pleca acasa fericit ☺

Episodul al paisprezecelea– Silverlight

Protagonistul nostru si-a dat seama ca a inceput sa invete XAML asa ca i-a venit o idee: „Daca tot a incercat sa faca aplicatia in WPF, de ce nu ar face-o sa mearga si in browser?”. Asa ca a dat un search in motorul lui de cautare preferat si a aflat ca fratele mai mic al lui WPF este Silverlight. Nici nu a stat pe ganduri si a mers pe www.silverlight.net si a dat peste o gramada de documentatie, exemple, si cel mai important pentru el in momentul respestiv, cum sa il instaleze.

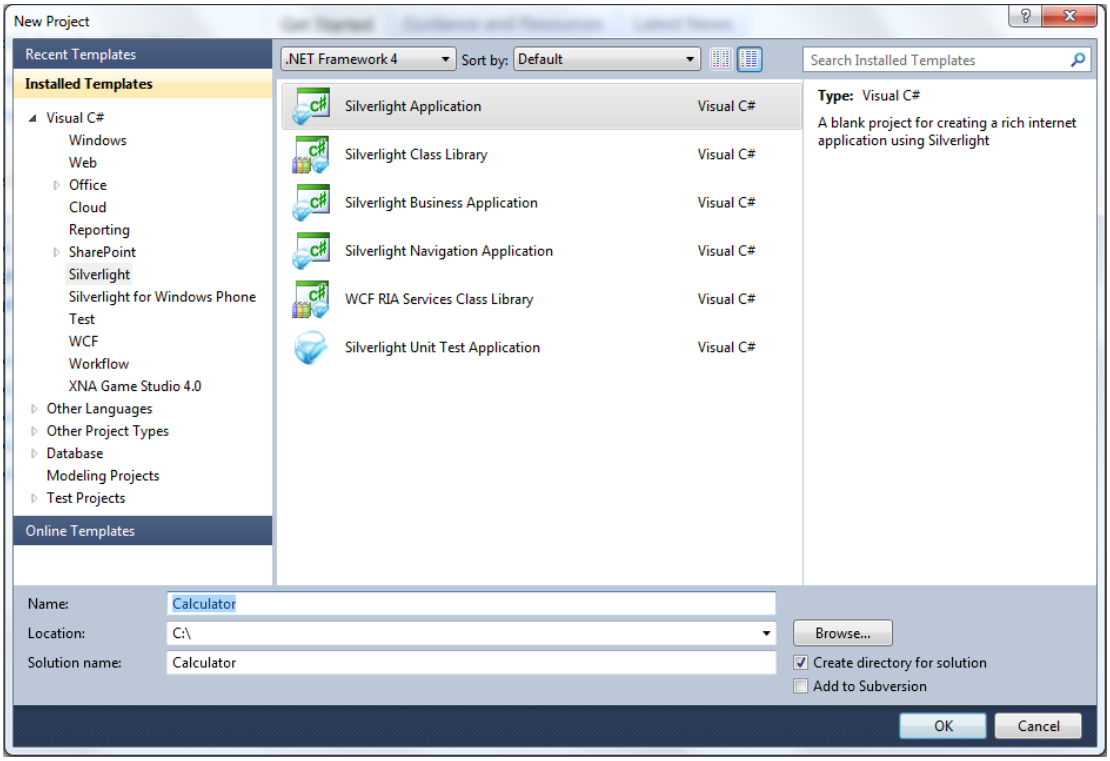
A fost si mai incantat cand si-a dat seama ca poate sa instaleze tot ce are nevoie despre Silverlight cu un singur installer - http://go.microsoft.com/fwlink/?LinkID=177428 asta fiindca avea Visual Studio 2010 instalat.

http://ronua.ro/src=babysteps

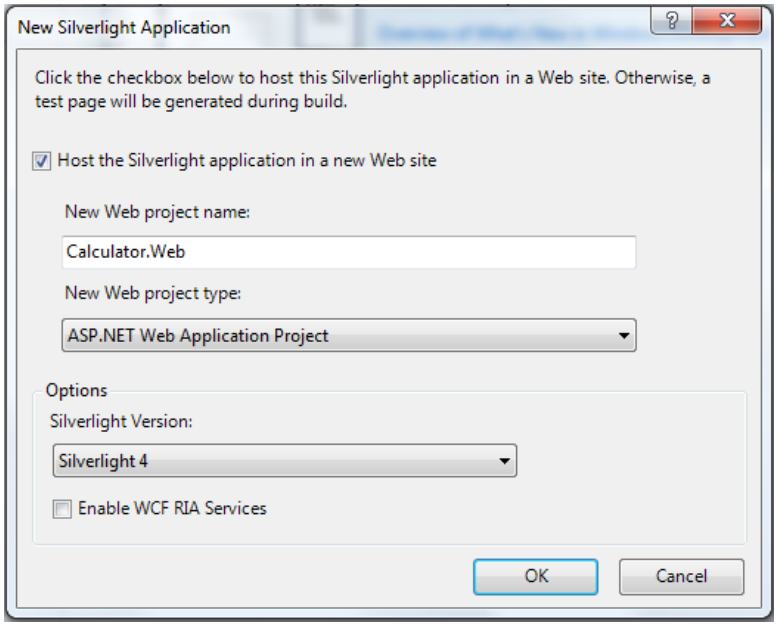


Dupa ceva timp petrecut pe acest site s-a apucat de treaba.
Stia deja despre MVVM si despre servicii WCF asa ca s-a hotarat sa le puna in aplicare. Intr-un final a ajuns la concluzia ca nu are nevoie de partea de server pentru aplicatia sa deoarece operatia de adunare se poate face si pe client si astfel user-ul o sa fie mai multumit de timpul de raspuns mic.
In urmatoarele ore a parcurs urmatorii pasi:

A creat o noua aplicatie Silverlight 4 .NET 4 in Visual Studio 2010;

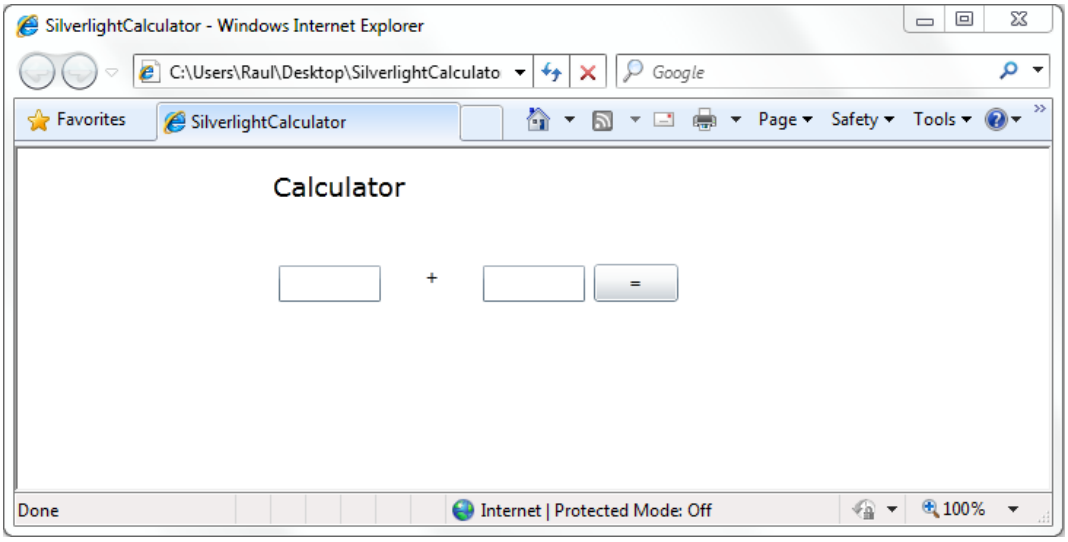


A ales sa hosteze aplicatia sa intr-un nou Web site;



Si astfel a facut prima sa aplicatie Silverlight 4.
Fiind incantat ca poate sa foloseasca aceleasi controale pe care le-a folosit si la aplicatia pentru Windows Phone 7 si cea de WPF, a ales sa construiasca design-ul.

Zis si facut! Dupa ce a adaugat cateva 2 TextBlock-uri , 2 TextBox-uri si un Button aplicatia lui arata astfel:



Iar codul sursa al aceste interfete este:

```
    <Grid x:Name="LayoutRoot" Background="White" Height="100" Width="350" >
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
            <ColumnDefinition/>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <TextBox Grid.Column="0"
                 Height="25" TextAlignment="Right"/>
        <TextBox Grid.Column="2"
                 Height="25" TextAlignment="Right"/>
        <TextBlock Grid.Column="4"
                   Height="25" TextAlignment="Left"/>
        <Button Margin="6,12"
                Content="=" Grid.Column="3" />
        <TextBlock Grid.Column="1" Text="+" Height="25"
TextAlignment="Center"/>
        <TextBlock Grid.ColumnSpan="3" Height="39" HorizontalAlignment="Left"
                   Margin="-4,-54,0,0" Text="Calculator"
                   VerticalAlignment="Top" Width="156" FontSize="18" />
    </Grid>
```

Dupa ce a trecut deja prin realizarea a o serie de aplicatii acum si-a facut o lista cu lucrurile importante pe care trebuie sa le ia in considerare atunci cand incepe sa faca arhitectura unei apliatii:

- Sa nu aiba cod de logica in fisierul xaml.cs, adica in fisierul de cod din spatele fisierului xaml deoarece este greu de intretinut si de testat;
- Trebuie sa realizeze un model astfel incat sa fie usor de testat;
- Trebuie sa organizeze resursele astfel incat acestea sa fie reutilizabile;
- O aplicatie trebuie sa ia in considerare validatoare.

Deoarece deja a implementat aplicatia in WPF si a folosit MVVM ca si pattern, Popescu vrea sa faca acelasi lucru si in Silverlight.

Se documenteaza putin si vede ca poate sa aplice comenzile si in Silverlight, doar ca trebuie sa le creeze el.

Fiind lucrul cel mai nou se apuca sa il faca. Desi se apropia ora plecarii s-a apucat sa impementeze o noua comanda si a si reusit sa faca asta inainte sa plece. Inca o zi in care Popescu pleca fericit acasa.

Iata si clasa realizata cu ajutoru articolului lui John Smith de pe MSDN “http://msdn.microsoft.com/en-us/magazine/dd419663.aspx” :

```
public class MyCommand : ICommand
    {
        private Action _handler;
        public MyCommand(Action handler)
        {
            _handler = handler;
        }

        private bool _isEnabled;
        public bool IsEnabled
        {
            get { return _isEnabled; }
            set
            {
                if (value != _isEnabled)
                {
                    _isEnabled = value;
                    if (CanExecuteChanged != null)
                    {
                        CanExecuteChanged(this, EventArgs.Empty);
                    }
                }
```

```
            }
        }

        public bool CanExecute(object parameter)
        {
            return IsEnabled;
        }

        public event EventHandler CanExecuteChanged;

        public void Execute(object parameter)
        {
            _handler();
        }
    }
```

A doua zi dimineta Popescu a revenit la lucru cu mare pofta de munca deoarece vroia sa vada ca munca lui da rezultate. Asa ca s-a apucat de ViewModel. Asta a fost partea usoara deoarece a mai facut-o o data in WPF. Astfel are o clasa ViewModel care implementeaza INotifyPropertyChanged care arata cam asa:

```
public class MyViewModel : INotifyPropertyChanged
    {
        private int? _firstOperator;
        private int? _secondOperator;
        private int? _result;

        private readonly ICommand _calculateCommand;

        public event PropertyChangedEventHandler PropertyChanged;

        public MyViewModel()
        {
            _calculateCommand = new MyCommand(Calculate){IsEnabled = true};
        }

        private void Calculate()
        {
            Result = Convert.ToInt32(FirstOperator) +
Convert.ToInt32(SecondOperator);
        }

        public int? FirstOperator
        {
            get { return _firstOperator; }
```

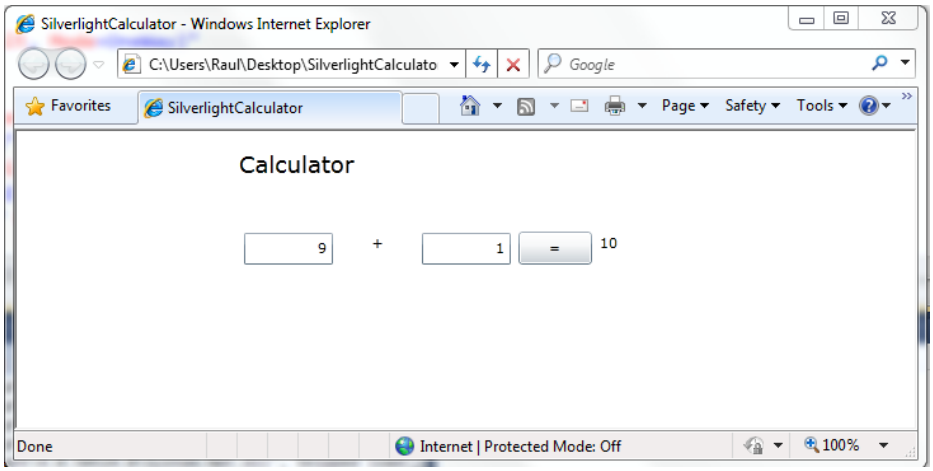
```
            set
            {
                _firstOperator = value;
                OnPropertyChanged("FirstOperator");
            }
        }

        public int? SecondOperator
        {
            get { return _secondOperator; }
            set
            {
                _secondOperator = value;
                OnPropertyChanged("SecondOperator");
            }
        }

        public int? Result
        {
            get { return _result; }
            private set
            {
                _result = value;
                OnPropertyChanged("Result");
            }
        }

        public ICommand CalculateCommand
        {
            get { return _calculateCommand; }
        }

        protected void OnPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this,
                    new PropertyChangedEventArgs(propertyName));
            }
        }
    }
```

A realizat si ViewModel-ul acum mai are sa il lege de View si in sfarsit o sa aiba ceva rezultare vizibile. Inca un lucru usor fiindca el deja stie sa faca Binding.

```
        <TextBox Grid.Column="0" Text="{Binding FirstOperator, Mode=TwoWay}"
```

```
            Height="25" TextAlignment="Right"/>
        <TextBox Grid.Column="2" Text="{Binding SecondOperator, Mode=TwoWay}"
            Height="25" TextAlignment="Right"/>
        <TextBlock Grid.Column="4" Text="{Binding Result, Mode=OneWay}"
              Height="25" TextAlignment="Left"/>
        <Button Grid.Row="1" Grid.ColumnSpan="5" Margin="0,5,0,0"
            Content="Calculate" Command="{Binding CalculateCommand}"  />
```

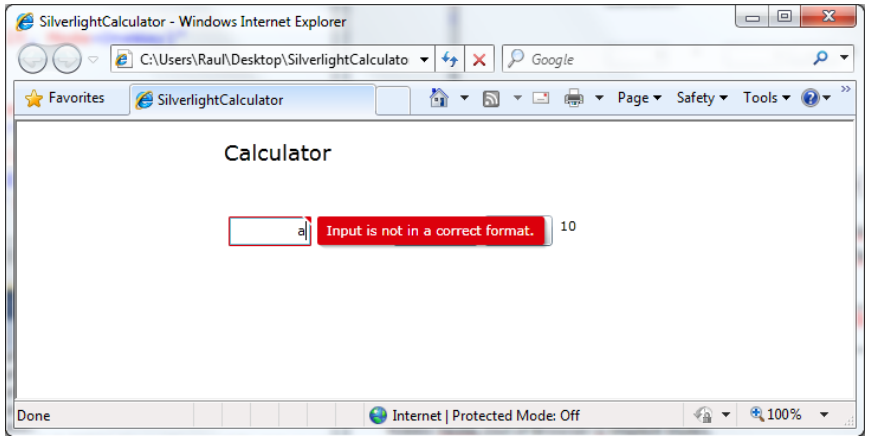
In sfarsit rezultate!



Se uita inca o data la lista pe care a facut-o initial si realizeaza ca inca nu a facut validari. Pentru asta mai are de adaugat un parametru la Binding. Astfel la fiecare TextBox Textul devine:

```
Text="{Binding FirstOperator, Mode=TwoWay, ValidatesOnExceptions=True}"
```

Si acum are si verificare:



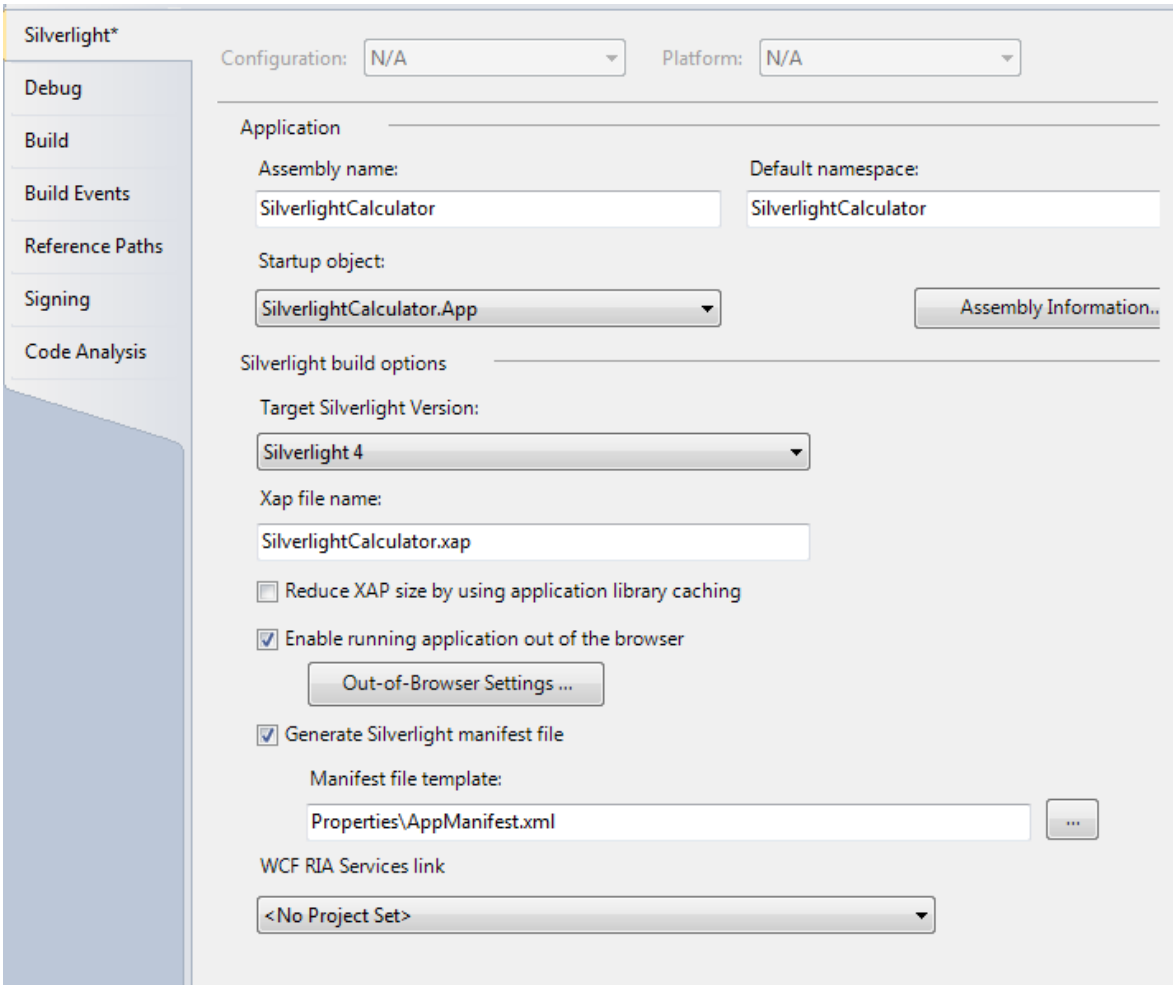
A fost foarte multumit de ce reusise pana acum, dar si-a amintit ca poate sa aiba custom errors. Si cu doar o mica modificare a celor 2 proprietati “Operator” a si reusit sa faca asta. Codul pe care l-a folosit este urmatorul:

```
        public string FirstOperator
        {
            get { return _firstOperator; }
            set
            {
                _firstOperator = value;
                try
                {
                    int.Parse(value.ToString());
                }
                catch (Exception)
                {
                    throw new Exception("This is not a number");
                }
                OnPropertyChanged("FirstOperator");
            }
        }
```

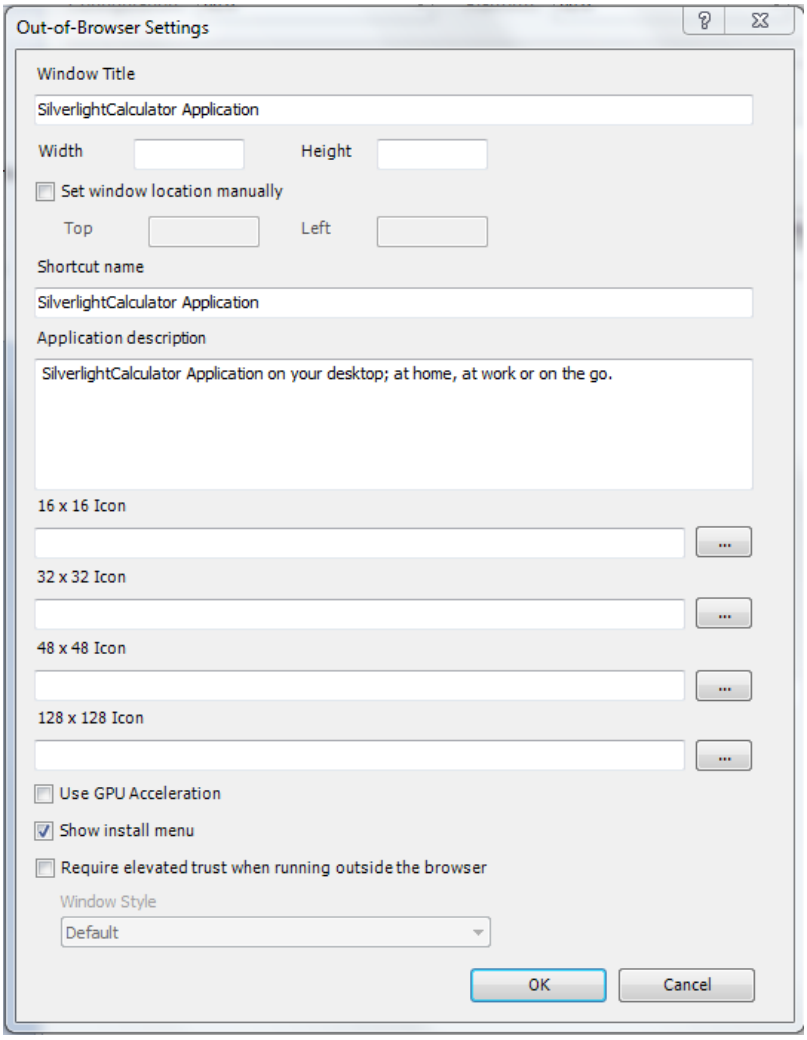
Desi avea deja o aplicatie Desktop, Popescu a vazut ca poate sa isi transforme aplicatia Silverlight in una Out of Browser.

A dar un search pe motorul de cautare preferat si a vazut ca nu are de facut decat cateva setari pentru a avea ceea ce si-a propus.

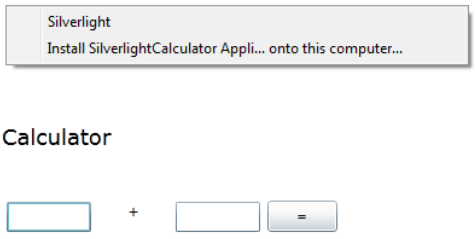
A dat click drapta pe proiectul de Silverlight si a mers la proprietati. Acolo a gasit proprietarea “Enable running application out of browser”.



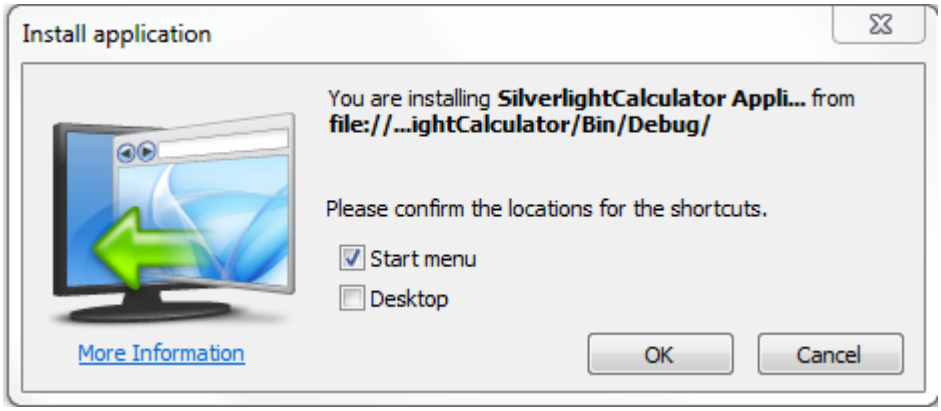
Dupa ce a facut asta a vazut ca poate sa seteze anumite proprietati ale aplicatiei sale si a deschis urmatorul modal.



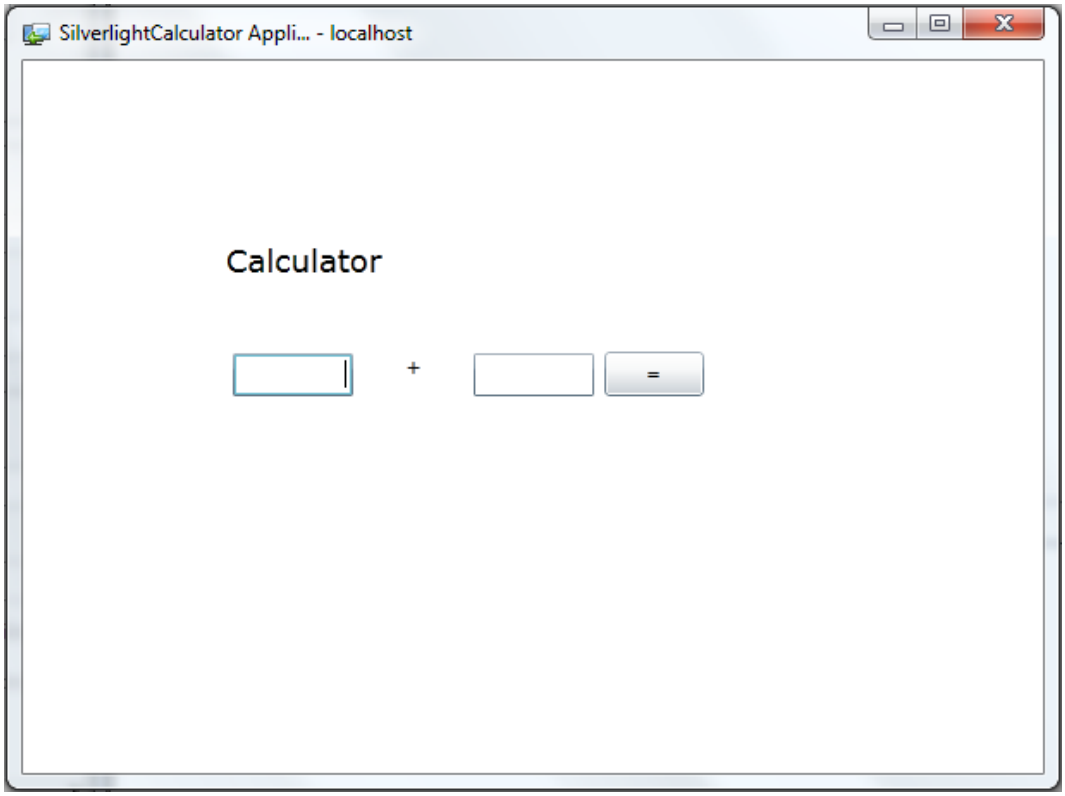
A urmat salvarea setarilor si dupa F5. Nu mica i-a fost mirarea cant a vazut ca atunci cand a dat click dreapta in browser mai avea inca o optiune.



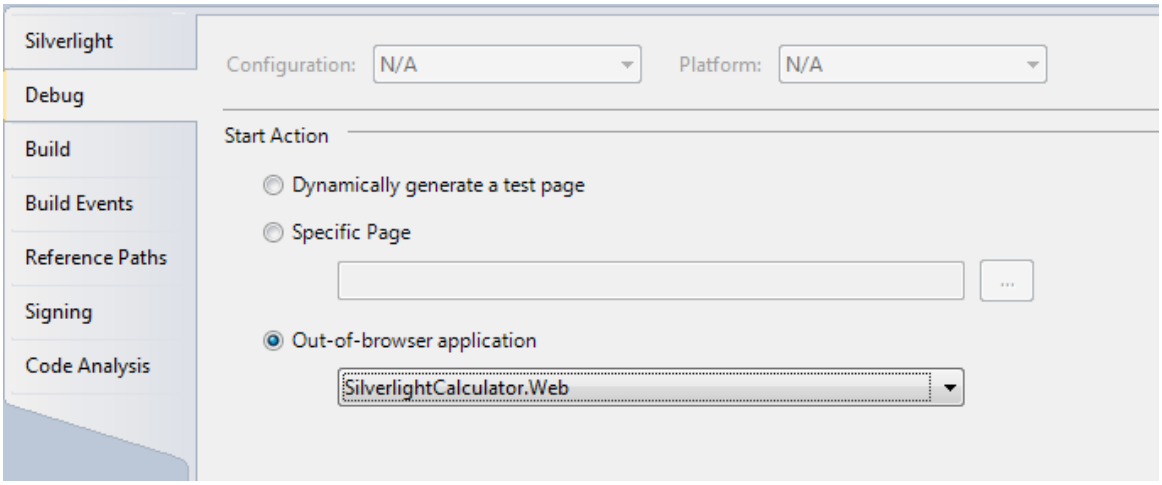
A dat click pe “Install…” si astfel a dat peste urmatorul modal.



Si astfel a fost doar la un click distanta de aplicatia lui Out of Browser.



Si asa Popescu si-a mai indeplinit inca o dorinta. Dar nu inainte de a face enable debugging si pentru aplicatia Out of Browser, asta tot din tabul de proprietati al aplicatiei Silverlight:

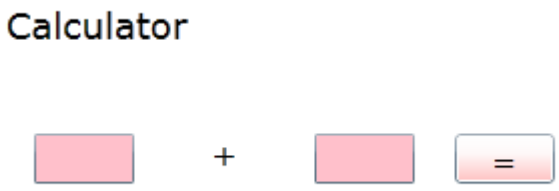


Acum daca tot avea aplicatia si in Browse si pe Desktop s-a gandit ca poate ar fi bine sa o si stilizeze un pic. Din videourile pe care le-a vazut a sesizat ca Silverlight ofera un mecanism de “Implicit styles”. A mai citit un pic despre asta si a vazut ca poate rezolva task-ul foarte usor, cu un scurt cod xaml scris in app.xaml.

De fapt care e treaba cu stilurile astea implicite: atunci cand setezi un stil pentru un anumit control, fiecare control de tipul respective va avea aceeasi stil fara a fi nevoie de specificare la fiecare control in parte.

A gasit un exemplu la “http://www.silverlightshow.net/items/Implicit-Styles-in-Silverlight-4.aspx” si l-a pus in aplicare pt fiecare element pe care l-a folosit.

Popescu era foarte multumit de ce facea in ziua aia asa a ajuns sa aiba o interfata destul de roz. Interfata lui arata cam asa:



Iar codul folosit a fost urmatorul:

```
<Style TargetType="Button">
            <Setter Property="Background" Value="Red"/>
            <Setter Property="Foreground" Value="Black"/>
            <Setter Property="FontSize" Value="16"/>
            <Setter Property="Height" Value="25"/>
            <Setter Property="Width" Value="50"/>
```

```
            <Setter Property="Margin" Value="8"/>
        </Style>
        <Style TargetType="TextBox">
            <Setter Property="Background" Value="Pink"/>
            <Setter Property="Foreground" Value="Black"/>
            <Setter Property="FontSize" Value="16"/>
            <Setter Property="Height" Value="25"/>
            <Setter Property="Width" Value="50"/>
            <Setter Property="Margin" Value="8"/>
        </Style>
        <Style TargetType="TextBlock">
            <Setter Property="Foreground" Value="Black"/>
            <Setter Property="FontSize" Value="16"/>
            <Setter Property="Height" Value="25"/>
            <Setter Property="Width" Value="50"/>
            <Setter Property="Margin" Value="8"/>
        </Style>
```

Acum in sfasit are ce si-a dorit. Are o aplicatie care respecta pattern-ul MVVM, care este atat In cat si Out of Browser pe care a reusit sa o si stilizeze. Acum e multimit, poate sa plece linistit acasa si maine sa revina la lucru pentru a primi urmatorul task.

Episodul al cincisprezecelea– alte lucruri interesante si enjoy the trip!

Astept sugestii de la voi pentru continuarea episoadelor

Nu am vorbit despre :

TODO VS2010 : Silverlight, Azure, Reporting , Sharepoint, DatabaseProject, SetupProject, Extensibility, TestProject, CodeAnalysis,Help, Ajax cu Jquery ,MEF, etc

TODO .NET 4 : CodeContracts, Parallel Extensions, (contra)variance, Tuples, SortedSet ,BigInteger

Sper ca v-a placut si va astept cu sugestii si participare!

Daca vreti sa participati sau sa primiti urmatoarele versiuni, va rog sa imi trimiteti email la ignatandrei@yahoo.com.

Va multumesc

Andrei Ignat