WComp : How to create a Bean ?

1 Contents

2 TUTORIAL

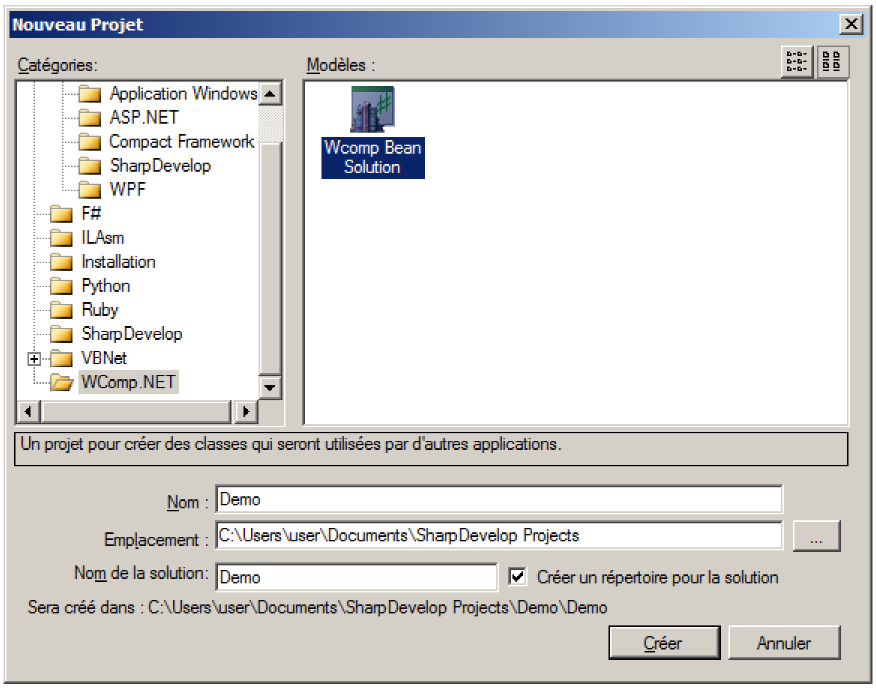
Let's start the tutorial. The first example shows the creating bean process. The second one shows how to create a simple bean. And, the third one is more complicated. It shows many bean capabilities.

The purpose of this tutorial is to show how to implement event programming in a bean.

2.1 CREATE AND USE A BEAN

2.1.1 FIRST STEP : CREATE A PROJECT

To create a WComp project, you must use File -> New -> Project/Solution



Un projet pour créer des classes qui seront utilisées par d'autres applications.

Nom : Demo

Emplacement : C:\Users\user\Documents\SharpDevelop Projects

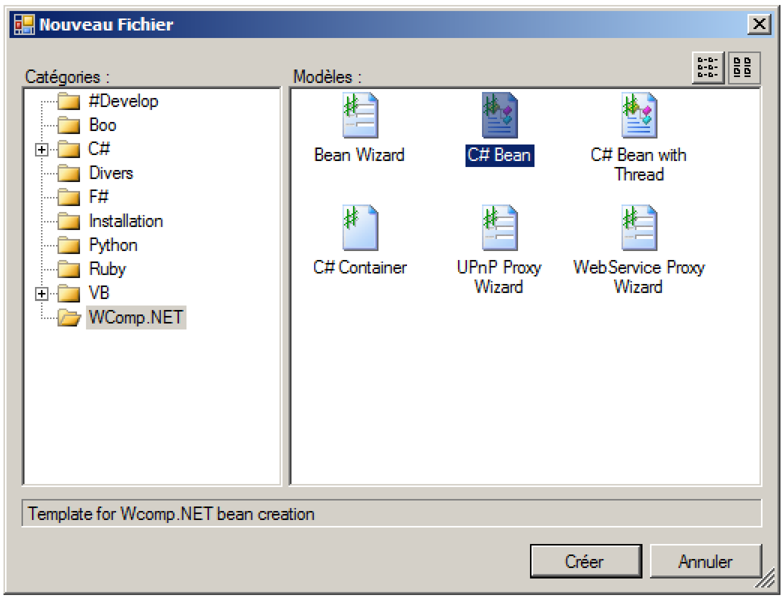
Nom de la solution: Demo ☑ Créer un répertoire pour la solution

Sera créé dans : C:\Users\user\Documents\SharpDevelop Projects\Demo\Demo

A dialog box appears, select WComp.NET -> WComp Bean Combine and enter the localisation and the name of your project. Then click on the "create" button.

2.1.2 SECOND STEP : CREATE A BEAN SOURCE FILE

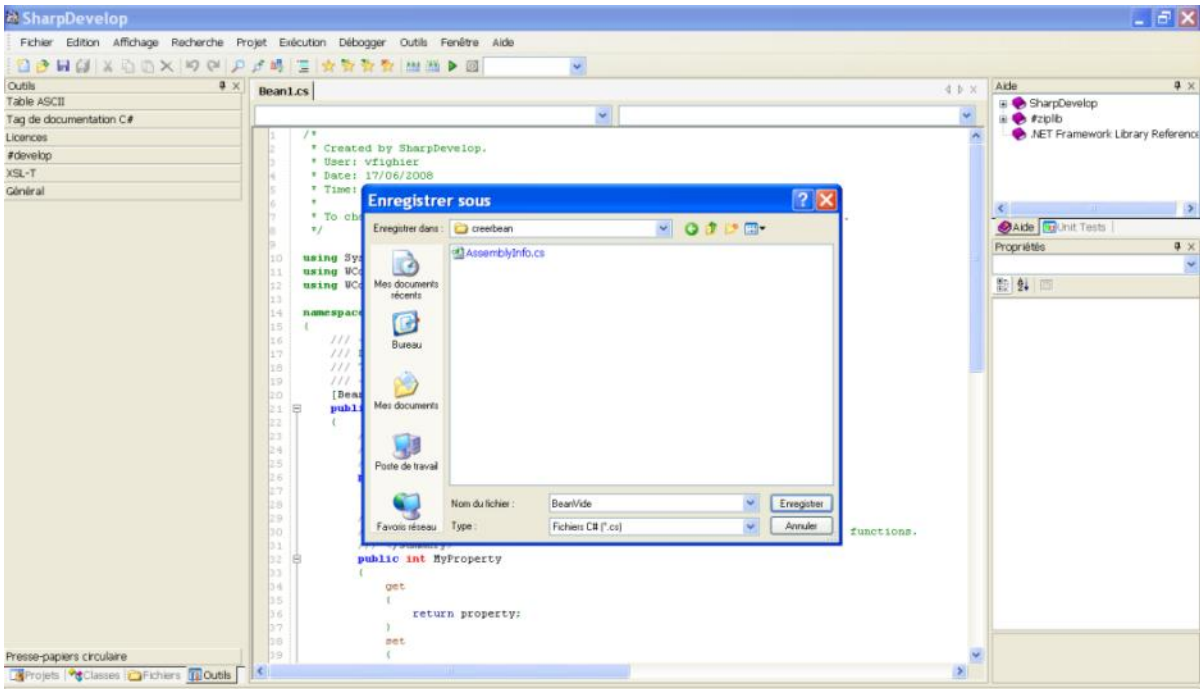
To create a WComp Bean file template, you must use File -> New -> File.



A dialog box appears, select WComp.NET -> C# Bean and click on the "create" button.

2.1.3 THIRD STEP : SAVE THE BEAN

To do it, you must use File -> Save As and choose the bean file name and its localisation.

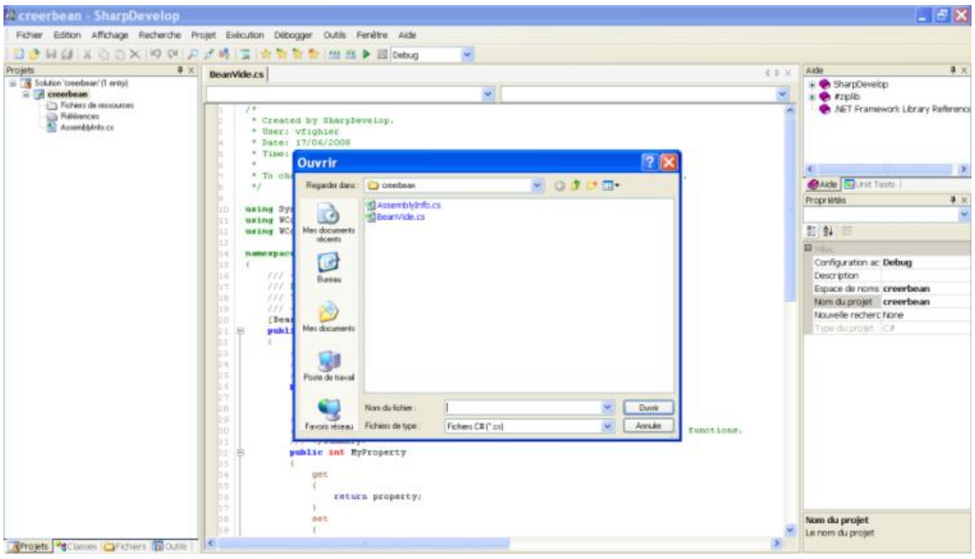


```
/*
 * Created by SharpDevelop.
 * User: vfighier
 * Date: 17/06/2008
 * Time:
 *
 * To ch
 */

using Sy
using WC
using WC

namespace
{
    ///
    ///
    ///
    ///
    [Bea
    publi
    {
                                                              functions.

        public int MyProperty
        {
            get
            {
                return property;
            }
            set
            {
```

A popup appears and indicate that the file has been saved.

2.1.4 FOURTH STEP : ADD THE BEAN TO THE PROJECT

In the "projects" slot, do a right click on your project. Then, select "Add" -> "Add files". A dialog box appears.



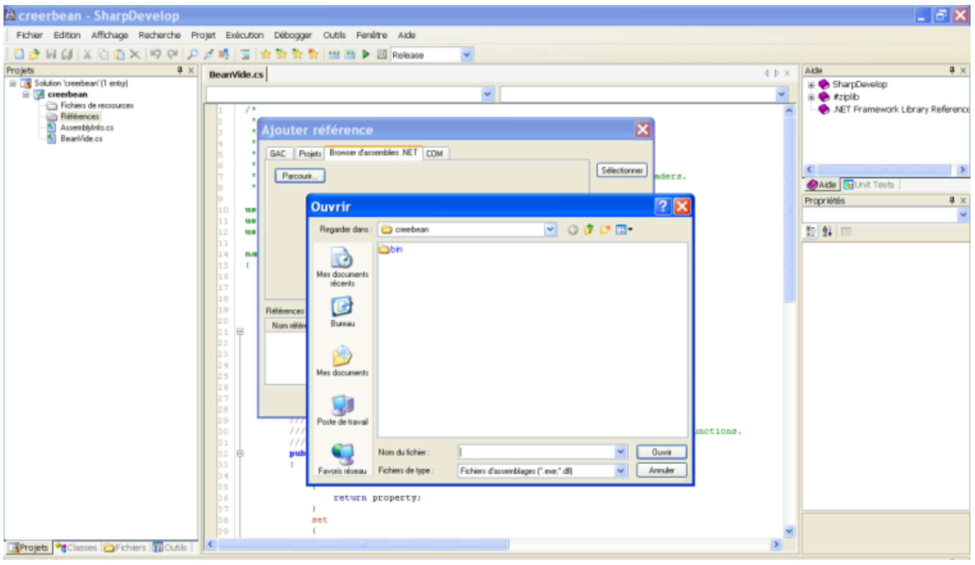
```
/*
 * Created by SharpDevelop.
 * User: vfighier
 * Date: 17/06/2008
 * Time:
```

```
public int MyProperty
    get
    {
        return property;
    }
    set
    {
```

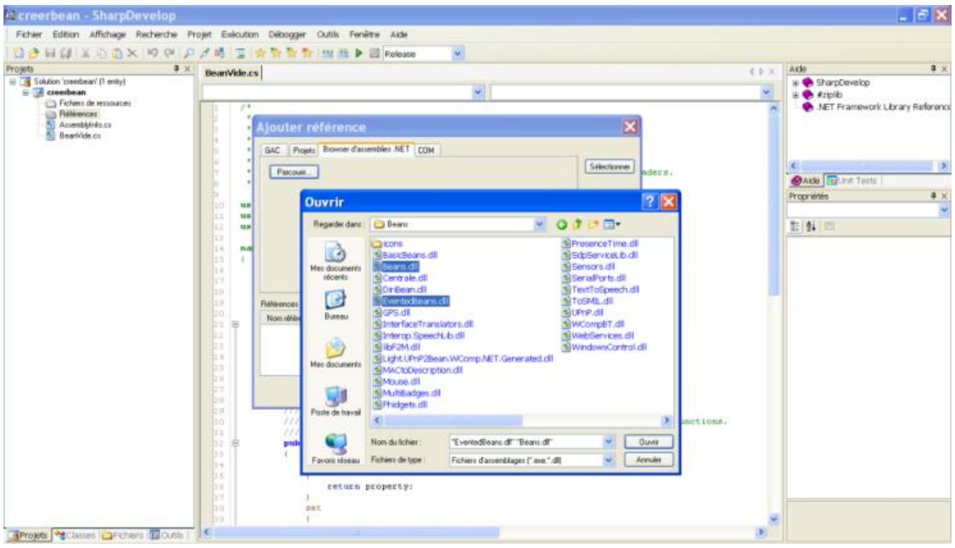
Select your bean file and press the “open” button. Your bean source file is added in your project.

2.1.5 FIFTH STEP : COMPILE YOUR BEAN

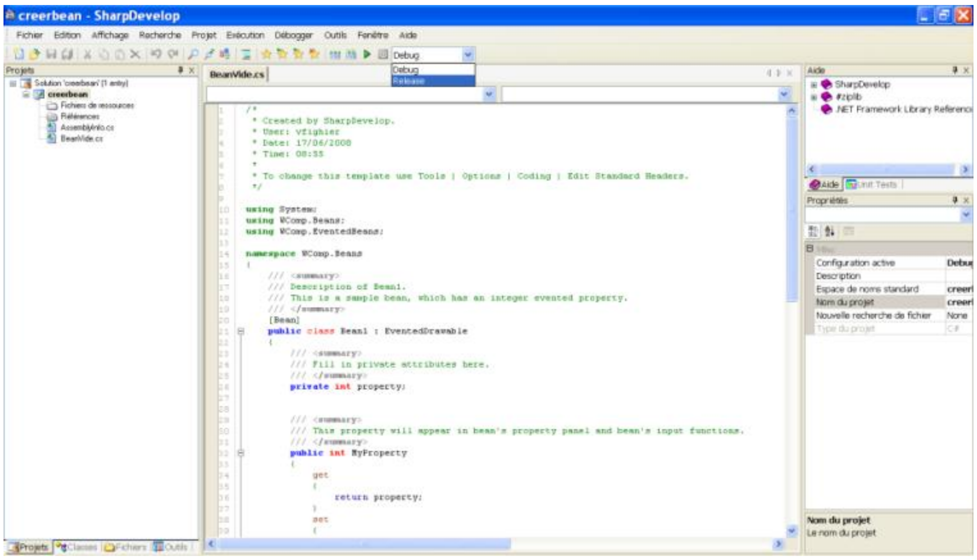
Firstly, In order to compile the bean, we must import some Dynamic Link Library (DLL). To do it, right click on the folder “references” of the project in the “projects” slot. And select “Add a reference”. A dialog box appears and in the slot “.NET assemblies browser” select “browse” button. A new dialog box appears.



Then, select WComp assemblies : “Beans” localised in “Beans” SharpDevelop folder.



Click on “Open” and then “Ok” buttons. In order to use a new bean in a WComp container you have to create a Dynamic Link Library (DLL) of it. To do that, you must compile your project with the “realise” mod. In the tool bar select the compile mod “release”.

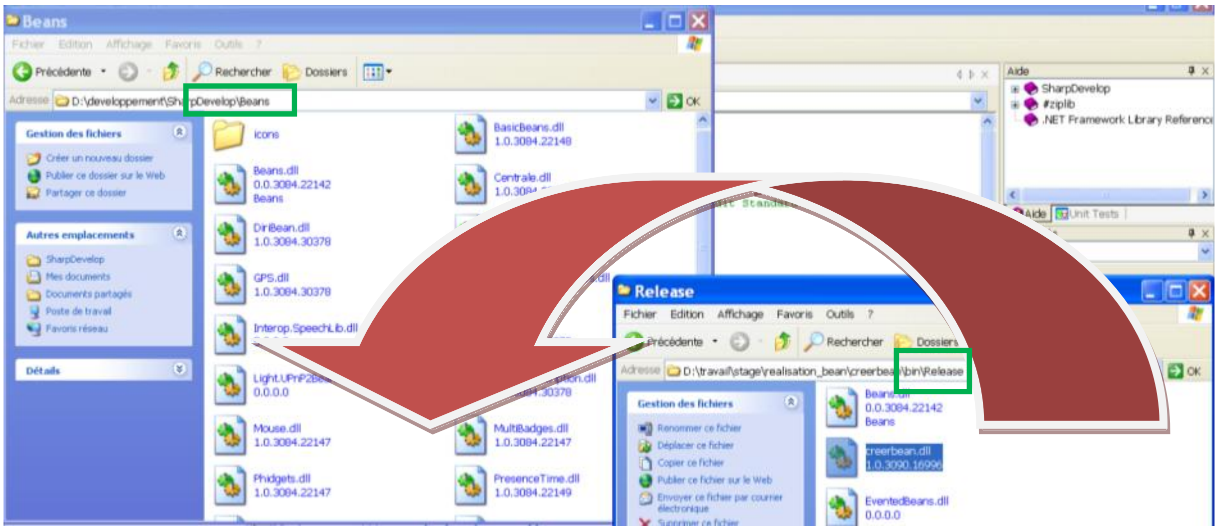


```
/*
 * Created by SharpDevelop.
 * User: vfighier
 * Date: 17/06/2008
 * Time: 08:55
 *
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using WComp.Beans;
using WComp.EventedBeans;

namespace WComp.Beans
{
    /// <summary>
    /// Description of Bean1.
    /// This is a sample bean, which has an integer evented property.
    /// </summary>
    [Bean]
    public class Bean1 : EventedDrawable
    {
        /// <summary>
        /// Fill in private attributes here.
        /// </summary>
        private int property;


        /// <summary>
        /// This property will appear in bean's property panel and bean's input functions.
        /// </summary>
        public int MyProperty
        {
            get
            {
                return property;
            }
            set
            {
```

And finally, compile your project. To do it, select “Executing” -> “build XXX” where XXX is your project name. The compilation creates a DLL of the project in the folder “bin/release” of the project.

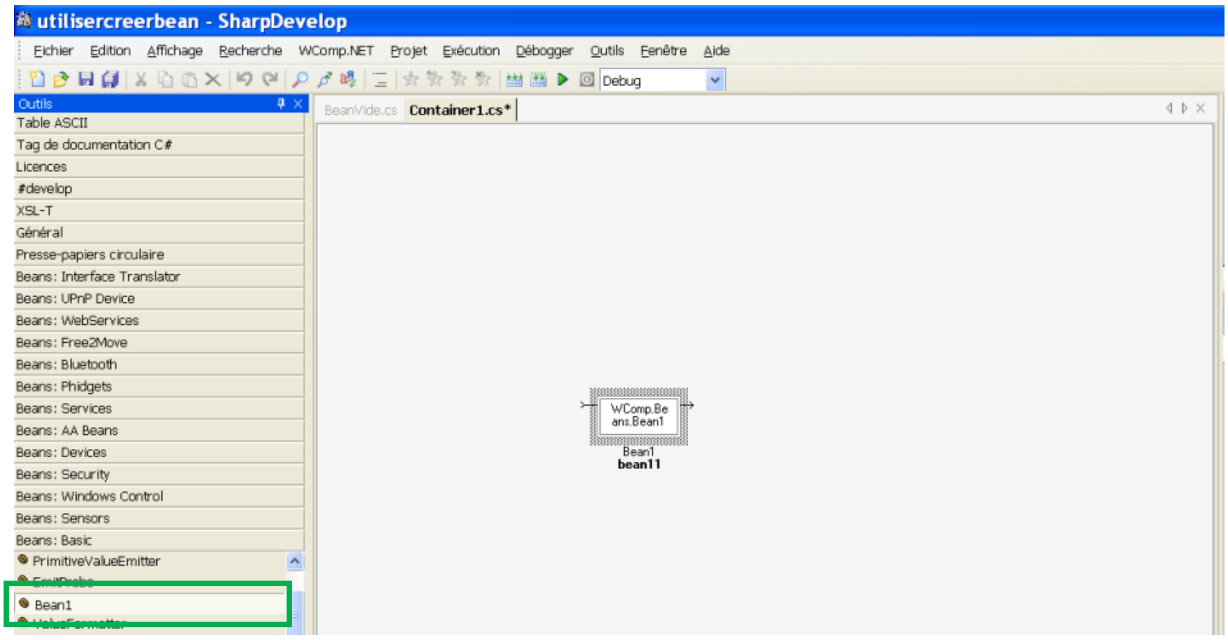
2.1.6 SIXTH STEP : ADD THE BEAN TO THE LIST

Now we have a compiled bean and we wants to use it in a WComp container. To do that, we must copy the bean DLL into SharpDevelop “Beans” folder.



2.1.7 SEVENTH STEP : USE THE BEAN IN ANOTHER PROJECT

You must restart SharpDevelop. Create a Container (File -> New -> File -> WComp Container). Add the bean to WComp container. In fact, the new bean appears into “Beans : Basic” into tools slot. Its name is its class name, in our case “Beans1”.



This new bean isn’t useful in WComp. In fact, this new bean is empty. Thus it doesn’t implement any functionality. We show how to add functionality in a bean in the next step.

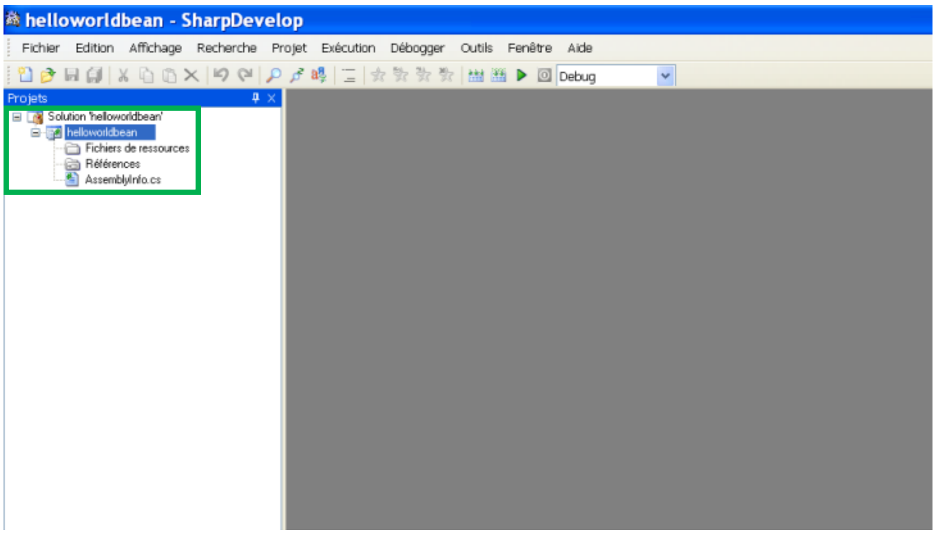
2.2 CREATE AND USE A SIMPLE BEAN

The WComp philosophy is that when you want some information you need to call a method. It riches an event and cause a call to a method which return some data. This example shows this mechanism.

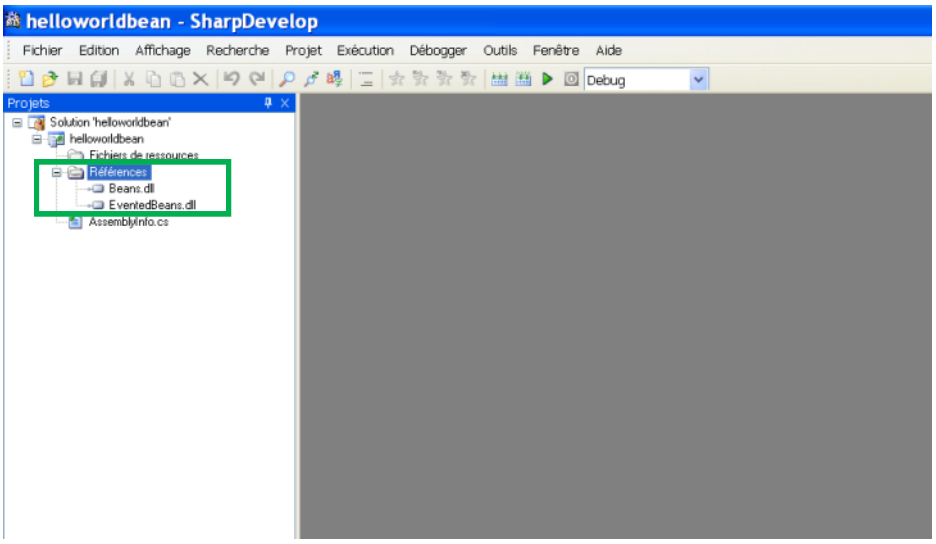
We will create a bean that returns the string “hello world” when the “FireHelloWorld” method is invoked.

2.2.1 FIRSTLY : CREATE THE NEW BEAN PROJECT.

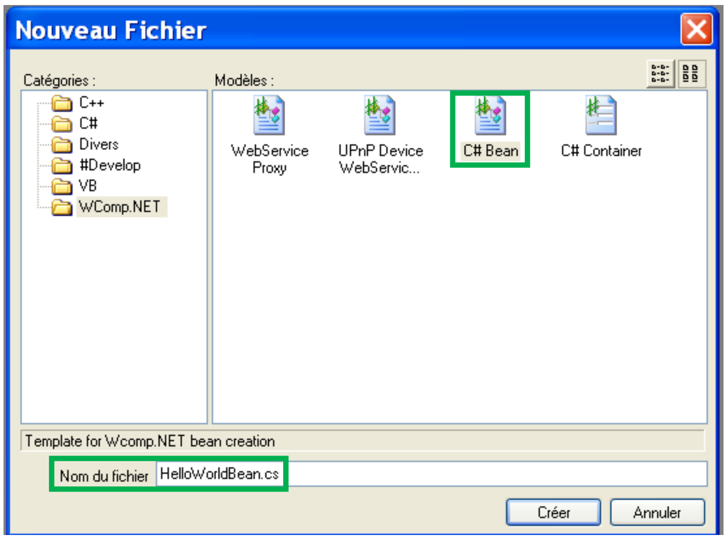
Firstly, we create a project called "helloworldbean". It contains the bean we want to create. The creation process was described previously.


helloworldbean - SharpDevelop
Fichier Edition Affichage Recherche Projet Exécution Débogger Outils Fenêtre Aide
Debug
Projets
Solution 'helloworldbean'
helloworldbean
Fichiers de ressources
Références
AssemblyInfo.cs

Add the two references (Beans and EventedBeans). The process was described previously.


helloworldbean - SharpDevelop
Fichier Edition Affichage Recherche Projet Exécution Débogger Outils Fenêtre Aide
Debug
Projets
Solution 'helloworldbean'
helloworldbean
Fichiers de ressources
Références
Beans.dll
EventedBeans.dll
AssemblyInfo.cs

Add the file into the project. To do it, right click on the project, select "Add" -> "New File". A Dialog box appears select "C# Bean". Then enter the bean name : "HelloWorldBeans".



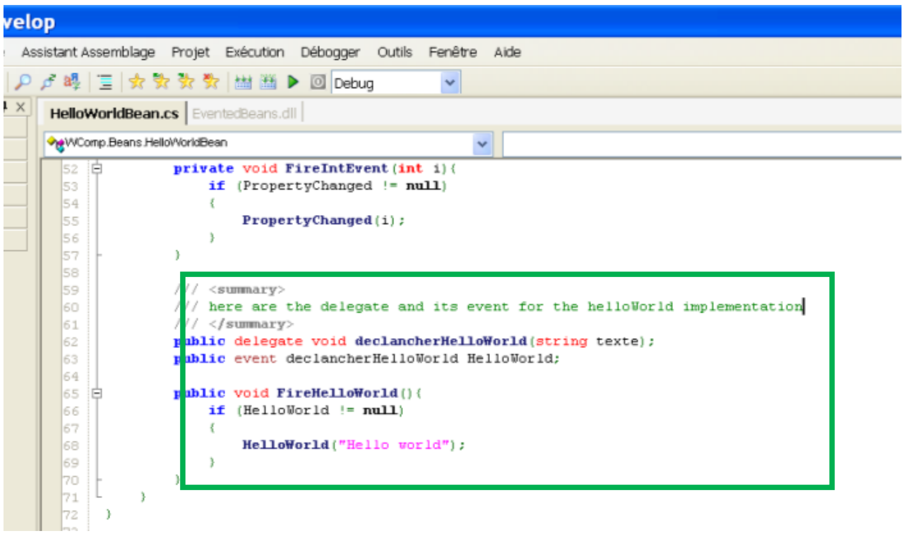
Template for Wcomp.NET bean creation

Nom du fichier: HelloWorldBean.cs

Click on button “Create”.

2.2.2 SECONDLY, IMPLEMENT THE NEW BEAN

Now, we want to add reactivity to our bean. We want to transmit the “hello world” string. Consequently, we need to rich an event that transmits the string. Thus we use a delegate. Moreover, we need to rich this event. So we use a method to do that. Here are codes to add to the bean source code.



```
        private void FireIntEvent(int i){
            if (PropertyChanged != null)
            {
                PropertyChanged(i);
            }
        }

        /// <summary>
        /// here are the delegate and its event for the helloWorld implementation
        /// </summary>
        public delegate void declancherHelloWorld(string texte);
        public event declancherHelloWorld HelloWorld;

        public void FireHelloWorld(){
            if (HelloWorld != null)
            {
                HelloWorld("Hello world");
            }
        }
    }
}
```

2.2.3 THIRDLY : COMPILE AND EXPORT NEW BEAN

Compile the project with release mode. Then copy the bean DLL into the SharpDevelop Beans folder.

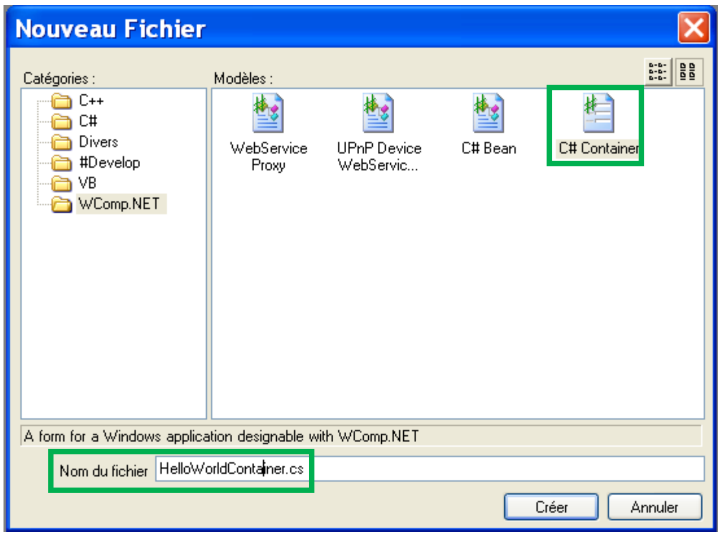
2.2.4 FOURTHLY : CREATE THE PROJECT WHICH USE THIS NEW BEAN

This is the same process to create a new project. The aim of this project is to use our new bean.

2.2.5 FIFTHLY : CREATE THE CONTAINER

First what is a container ? This is a WComp component that allows the user to add some beans and links between them. Consequently, a container allows us to build a complex component defined by beans and their behaviour.

To create a container, right click on the project select “Add” -> “New file”. A Dialog Box appears and select “WComp.NET” -> “C# Container”. Fill in the container name. In our case, we choose “HelloWorldContainer”.



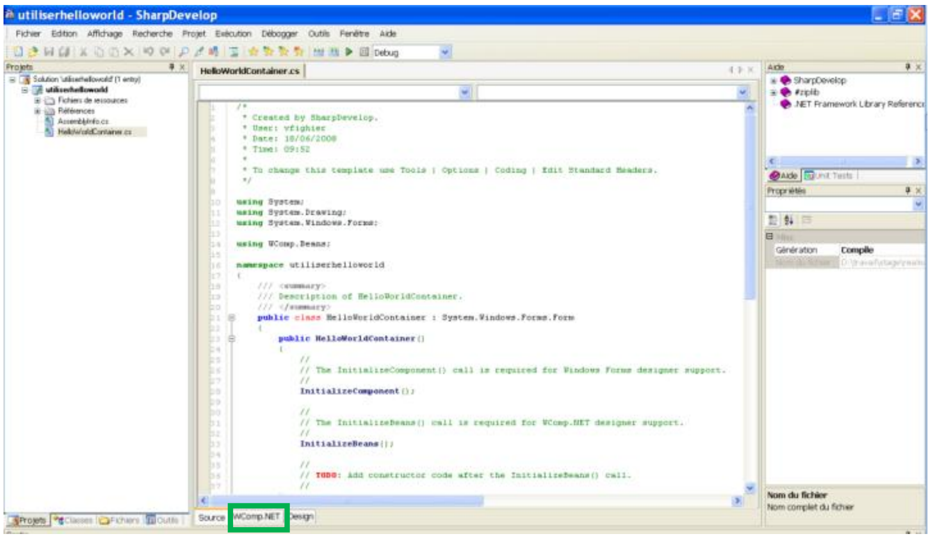
A form for a Windows application designable with WComp.NET

Nom du fichier: HelloWorldContainer.cs

Then click on the “create” button.

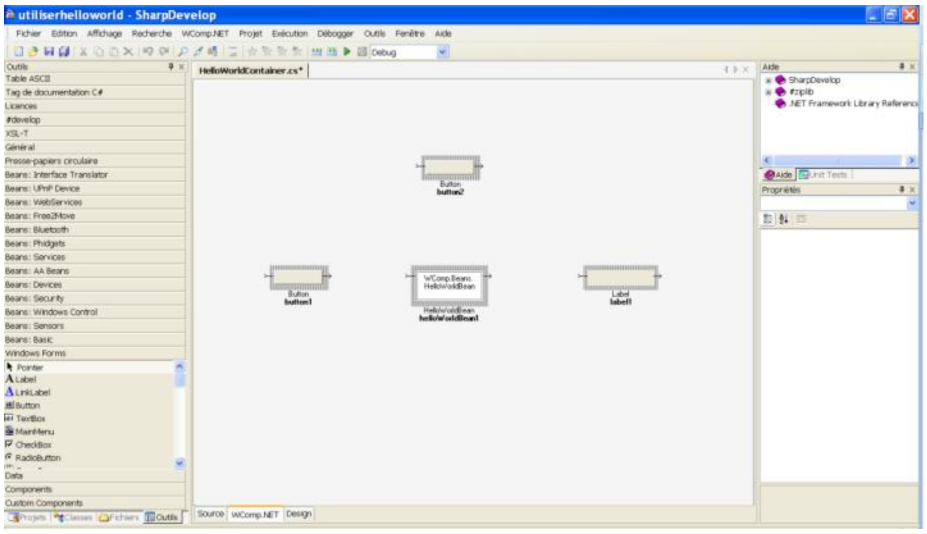
2.2.6 SIXTHLY : IMPLEMENT THE CONTAINER

The container is created. The source container code is displayed. We switch the display mode into WComp.NET mode.

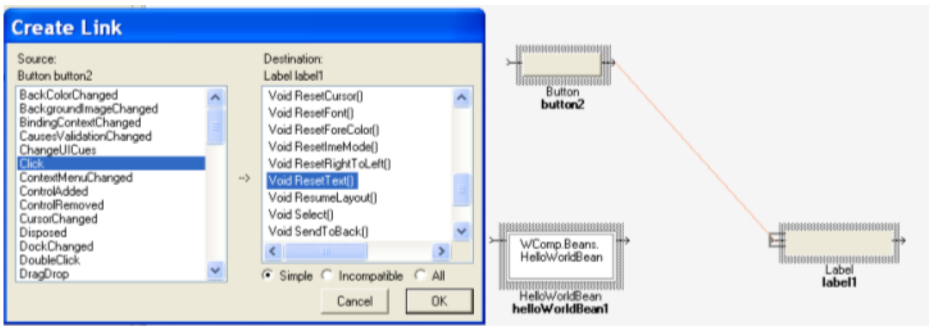


```
/*
 * Created by SharpDevelop.
 * User: vfighier
 * Date: 18/06/2008
 * Time: 09:52
 *
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */

using System;
using System.Drawing;
using System.Windows.Forms;

using WComp.Beans;

namespace utiliserhelloworld
{
	/// <summary>
	/// Description of HelloWorldContainer.
	/// </summary>
	public class HelloWorldContainer : System.Windows.Forms.Form
	{
		public HelloWorldContainer()
		{
			//
			// The InitializeComponent() call is required for Windows Forms designer support.
			//
			InitializeComponent();

			//
			// The InitializeBeans() call is required for WComp.NET designer support.
			//
			InitializeBeans();

			//
			// TODO: Add constructor code after the InitializeBeans() call.
			//
```

To this tutorial, we add four beans in the container : two buttons, a HelloWorldBean and a label. To add a component into the container, first, select the component into the tools section and click on the design part where you want to add the component.



First, we make that the button at the top erase the label text. To do it, select the button output and link it to the label input. A Dialog box appears, select the “Click” source and the “ResetText()” destination.

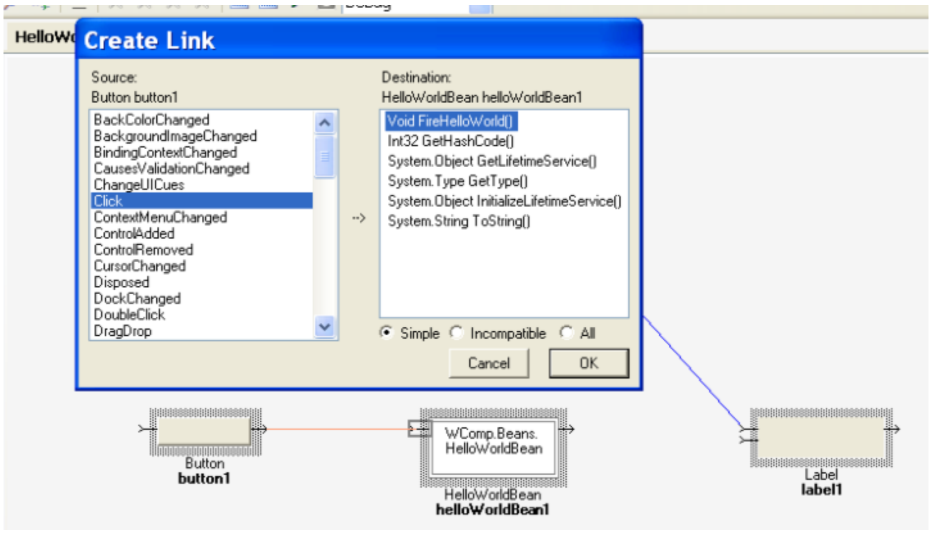

Create Link

Source:
Button button2

Destination:
Label label1

Then click on the “OK” button.

Then, we want to cause a helloWorld method call when we click on the button. To do it, we add a link between the second button and the helloWorld bean. In the dialog box select “Click” as source and the method “FireHelloWorld” as destination.

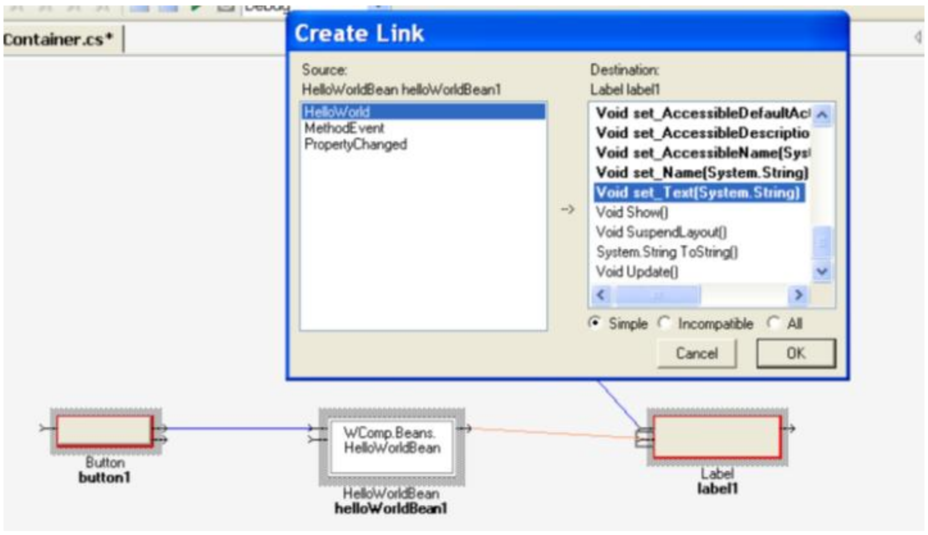

Create Link

Source:
Button button1

Destination:
HelloWorldBean helloWorldBean1

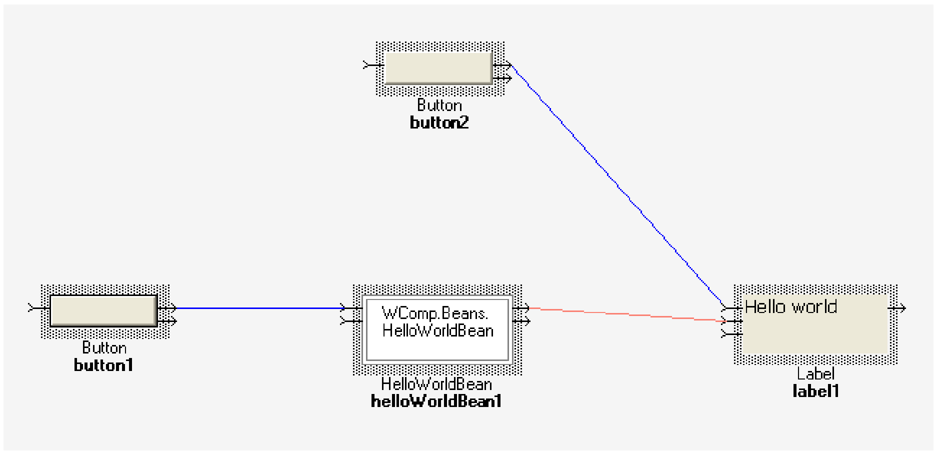
Then click on the “OK” button.

Finally, we want to display on the label the helloWorld call result. Thus, we add a new link between the HelloWorld bean and the label. Select "HelloWorld" as source and "set_Text" as destination, in the dialog box.



2.2.7 SEVENTHLY : TEST THE CONTAINER

Click on the button at the right, and "hello world" will be printed in the label.



Click on the button at the top, and the text in the label will disappear.

2.3 CREATE AND USE A COOL BEAN

We wanted to create a cool bean. It creates some random numbers. This numbers are in a range. It is bounded by a minimum and a maximum. Moreover we can choose the number of random number we want.

2.3.1 FIRST STEP : CONFIGURE THE PROJECT

First we create the bean project in order to implements the bean. We call it "coolbean". Then we add the two references (Beans and EventedBeans) of SharpDevelop bean folder. Then we add a new c# bean source file. We call it "coolbean".

2.3.2 SECOND STEP : IMPLEMENT THE BEAN

First, we want to customize the bean. Consequently, we implements three setters. Each one changes a piece of information of the bean :

– One changes the number of random numbers we want, made by the "setNb" method.
– One updates the minimum value of random numbers, made by the "setMin" method.
– One changes the maximum value of random numbers, made by the "setMax" method.

Thus the code to insert into the bean source code is :

```
public void setMin(string val){
        try{
                this.min = Int32.Parse(val);
        }
        catch{}
}

public void setMax(string val){
        try{
                this.max = Int32.Parse(val);
        }
        catch{}
}

public void setNb(string val){
        try{
                this.nb = Int32.Parse(val);
        }
        catch{}
}
```

Then, we want the bean to generate some random numbers. So we need a method to do that. Here is the implementation :

```
private int genererNombre( Random generateurAleatoire ){
        int interval = this.max - this.min + 1;
        int nombre = 0;

        nombre = generateurAleatoire.Next(interval);
        nombre += this.min;

        return nombre;
}
```

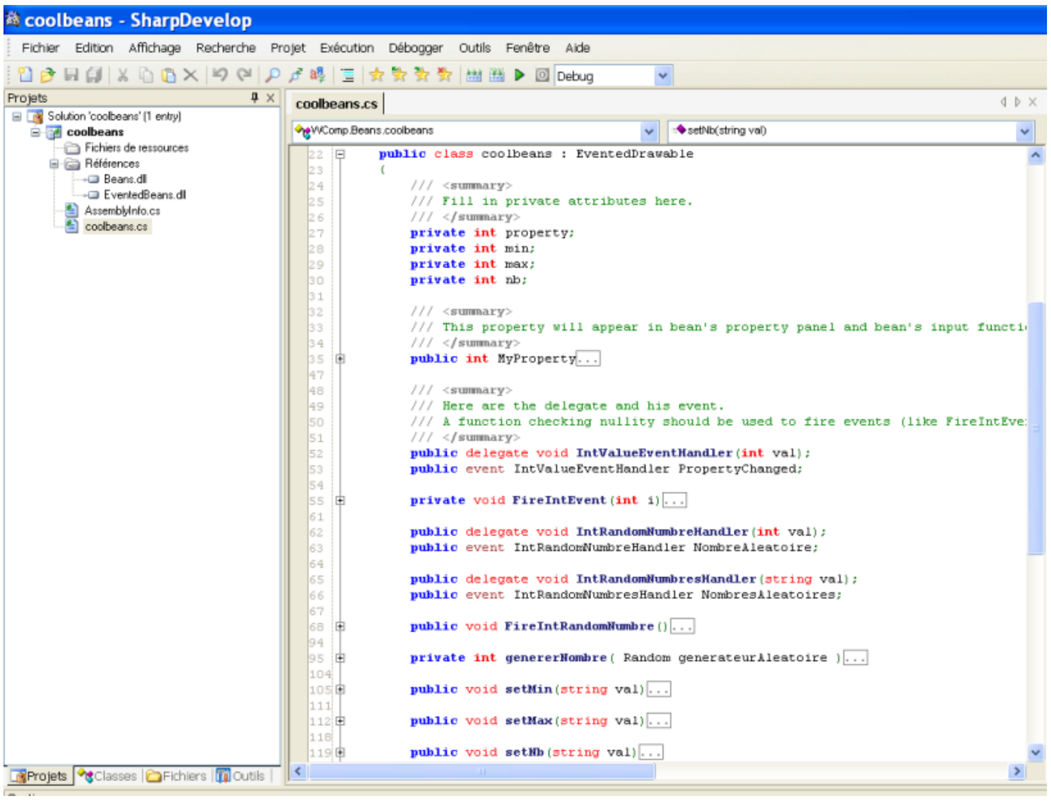
Then, we want the bean to generate some numbers. Our bean has two types of reaction. On one hand, when the user wants only one number. On the other hand, when he wants to generate several numbers. So we use two delegate methods to return the result. Thus the code to do that is :

```
public delegate void IntRandomNumbreHandler(int val);
public event IntRandomNumbreHandler NombreAleatoire;

public delegate void IntRandomNumbresHandler(string val);
public event IntRandomNumbresHandler NombresAleatoires;
```

Finally, to return some results using events, they have to be launched. Consequently, we need another method to do that. Its code is :

```
public void FireIntRandomNumbre(){
        Random generateurAleatoire  = new Random();
        StringBuilder nombres = null;
        int nombre = 0;
        int i = 0;

        if(nb == 1){
                nombre = this.genererNombre(generateurAleatoire);

                if( NombreAleatoire != null){
                        NombreAleatoire(nombre);
                }
        }else if( nb > 1 ){
                nombres = new StringBuilder();


                for(i=0 ; i<this.nb ; i++){
                        nombre = this.genererNombre(generateurAleatoire);
                        nombres.Append( ((Int32)nombre).ToString());
                        nombres.Append( "\n" );
                }

                if( NombresAleatoires != null ){
                        NombresAleatoires(nombres.ToString());
                    }
                }
            }
```

As a result, the project looks like :



```
    public class coolbeans : EventedDrawable
    {
        /// <summary>
        /// Fill in private attributes here.
        /// </summary>
        private int property;
        private int min;
        private int max;
        private int nb;

        /// <summary>
        /// This property will appear in bean's property panel and bean's input functi
        /// </summary>
        public int MyProperty...

        /// <summary>
        /// Here are the delegate and his event.
        /// A function checking nullity should be used to fire events (like FireIntEve
        /// </summary>
        public delegate void IntValueEventHandler(int val);
        public event IntValueEventHandler PropertyChanged;

        private void FireIntEvent(int i)...

        public delegate void IntRandomNumbreHandler(int val);
        public event IntRandomNumbreHandler NombreAleatoire;

        public delegate void IntRandomNumbresHandler(string val);
        public event IntRandomNumbresHandler NombresAleatoires;

        public void FireIntRandomNumbre()...

        private int genererNombre( Random generateurAleatoire )...

        public void setMin(string val)...

        public void setMax(string val)...

        public void setNb(string val)...
```

2.3.3 THIRD STEP : COMPILE AND EXPORT THE BEAN

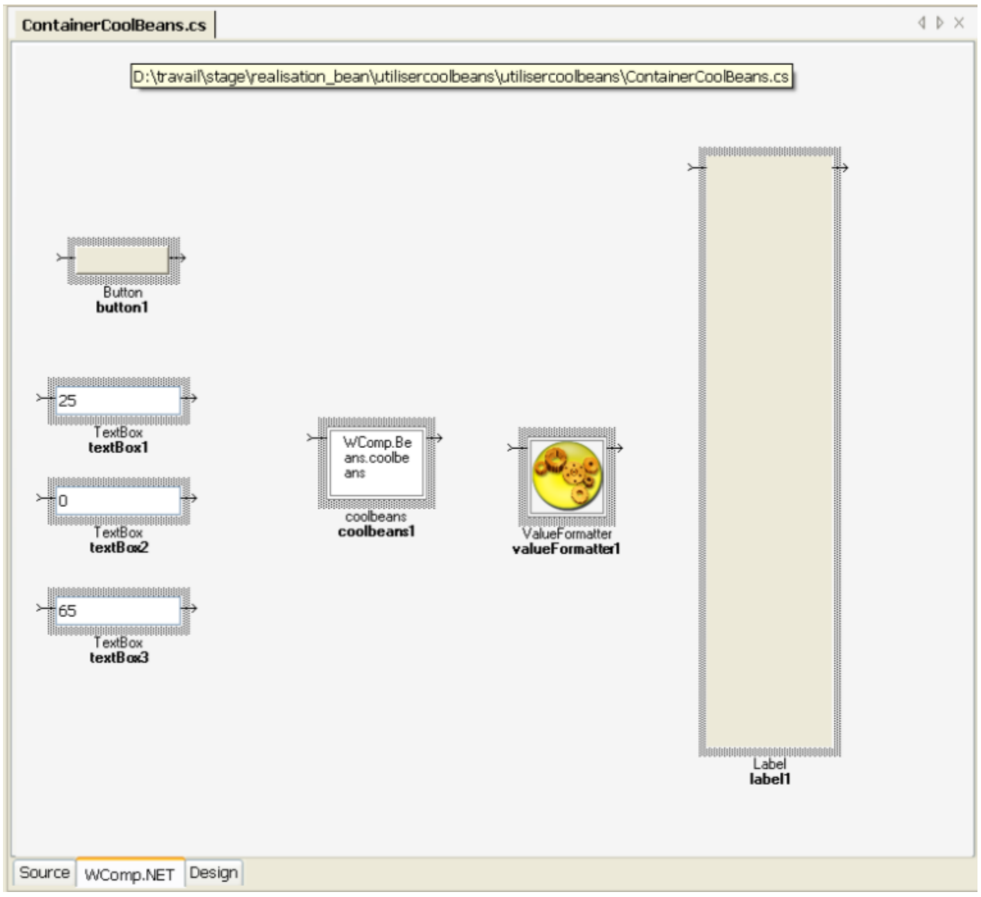
Now, the bean is implemented. We just need to compile it with the release mode and copy the DLL into SharpDevelop Bean folder. Then, we can use it into a new container.

2.3.4 FOURTH STEP : CREATE THE PROJECT WHICH USE THE BEAN

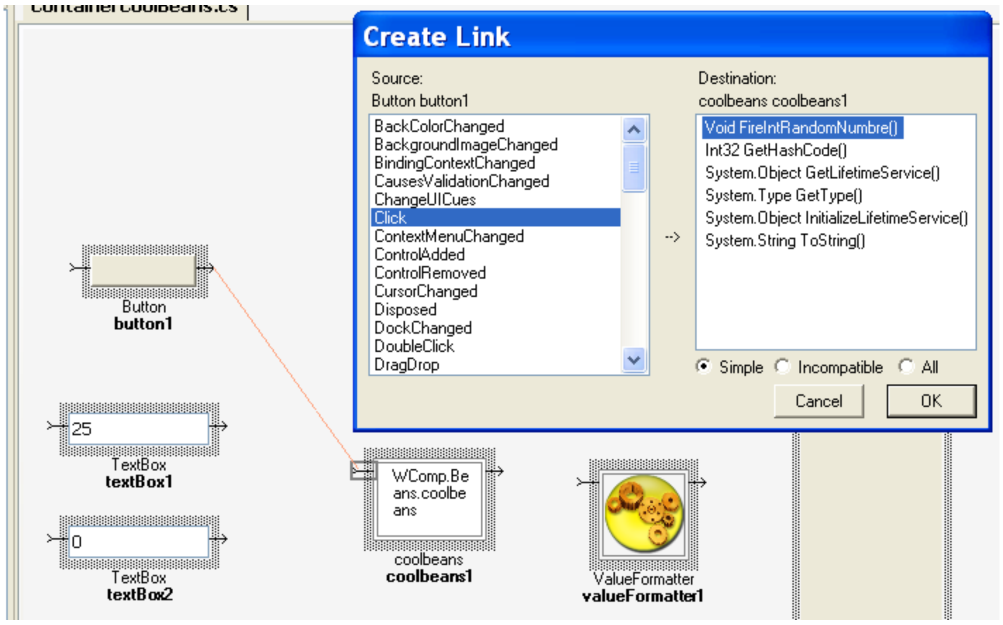
Now, it is time to use this bean. Thus we create a new project called “utilisercoolbeans”. Then add the two bean references. And create a new container file called “ContainerCoolBeans”

2.3.5 FIFTH STEP : IMPLEMENT THE CONTAINER

We need a coolbean component, three textbox, a button, a value formatter and a label.



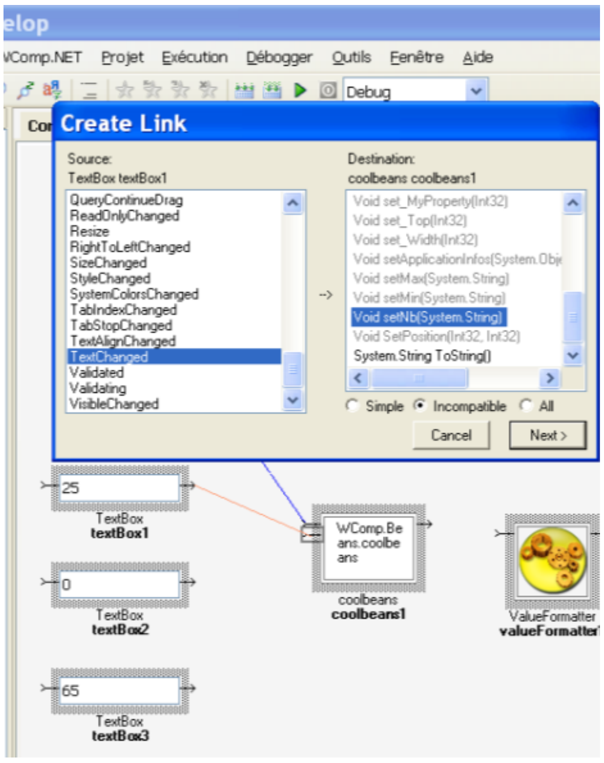
Then, we had to implement the component behaviours. First, we define the link that rich the numbers generation. That is to say the link between the button and the coolbean component.
To define this link, choose the “Click” source and “FireIntRandomNumbre” destination.



Source: Click
Destination: Void FireIntRandomNumbre()
[✓ Simple] Incompatible All

Then, we had to implement customisation part linked to the coolbean component. The textbox at the top sets the number of random number we want. The second textbox gives the minimum value of random numbers. And the last one, at the updates the maximum value of the number. The source used is “TextChange” and the destination is the corresponding setter with the value “get_Text”.

Here is an example of the edition of the link between a textbox, here the number of random numbers and the coolbean component.



Create Link

Source:
TextBox textBox1

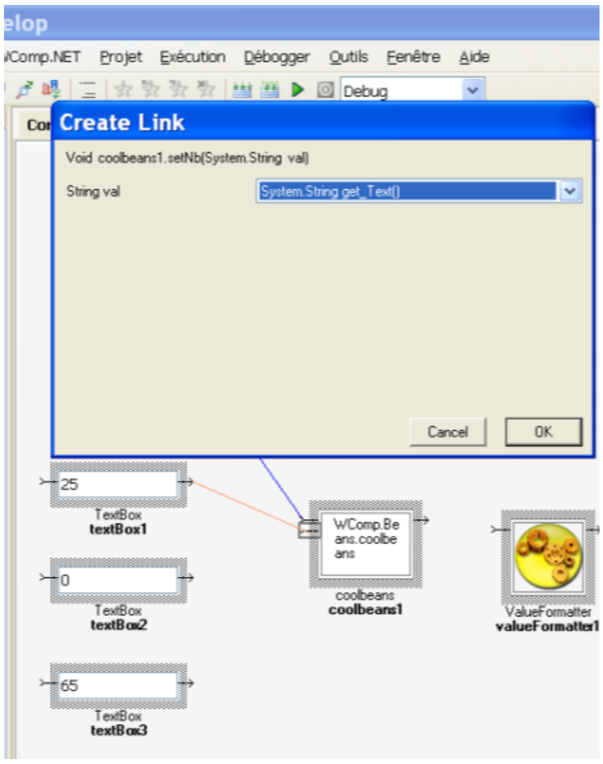
QueryContinueDrag
ReadOnlyChanged
Resize
RightToLeftChanged
SizeChanged
StyleChanged
SystemColorsChanged
TabIndexChanged
TabStopChanged
TextAlignChanged
TextChanged
Validated
Validating
VisibleChanged

-->

Destination:
coolbeans coolbeans1

Void set_MyProperty(Int32)
Void set_Top(Int32)
Void set_Width(Int32)
Void setApplicationInfos(System.Obj
Void setMax(System.String)
Void setMin(System.String)
Void setNb(System.String)
Void SetPosition(Int32, Int32)
System.String ToString()

Simple [✓ Incompatible] All

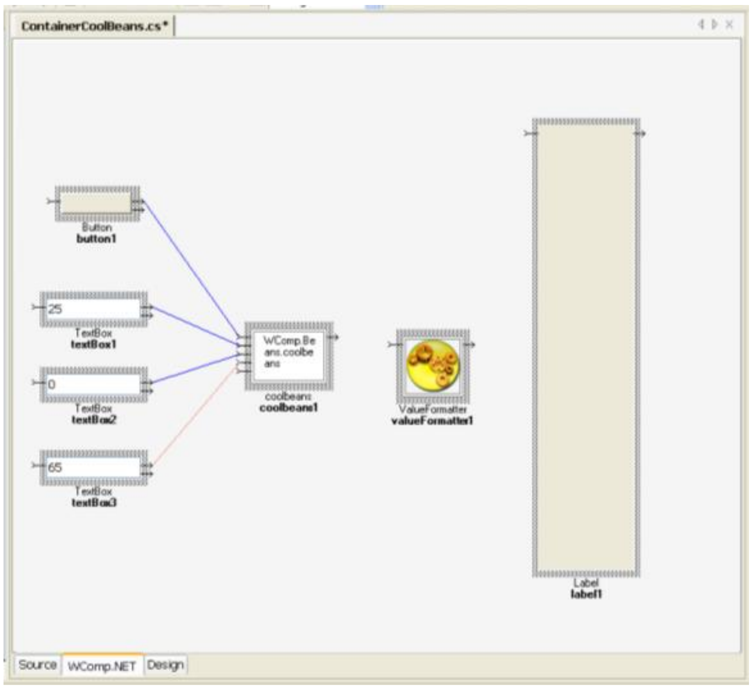


Create Link

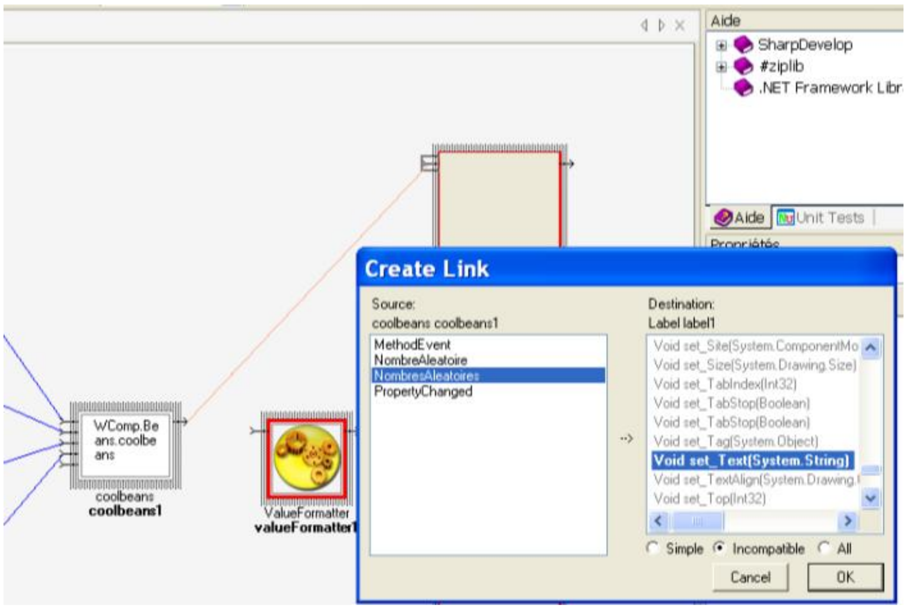
Void coolbeans1.setNb(System.String val)

String val System.String get_Text()

Use the same process to link the other textboxes to the coolbean component. We get this container



Now, with button and textboxes, we can rich events. However, we can’t see these results. And we will fill this lack. The first coolbean component event we will catch is called “NombresAleatoires”. It returns many random numbers in a string. Consequently, we just need to add a link between the coolbean component and the label. The source is “NombresAleatoires” and the destination is “set_Text”.



Create Link

Source:
coolbeans coolbeans1

MethodEvent
NombreAleatoire
NombresAleatoires
PropertyChanged

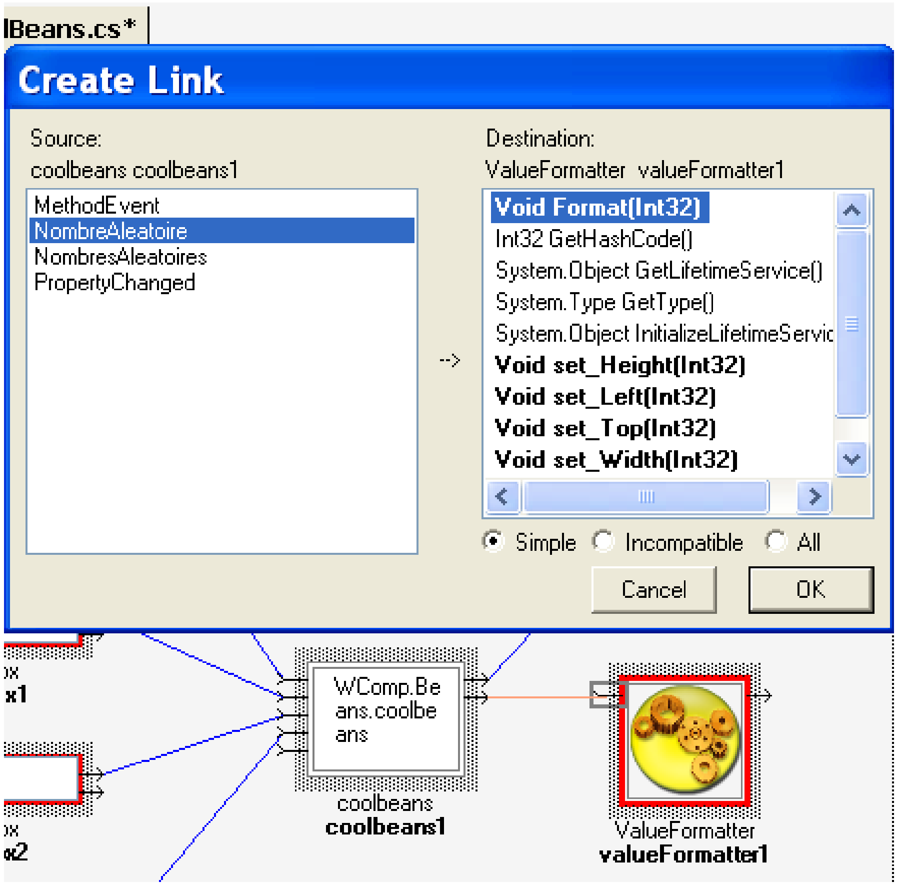
Destination:
Label label1

Void set_Site(System.ComponentMo
Void set_Size(System.Drawing.Size)
Void set_TabIndex(Int32)
Void set_TabStop(Boolean)
Void set_TabStop(Boolean)
Void set_Tag(System.Object)
Void set_Text(System.String)
Void set_TextAlign(System.Drawing.
Void set_Top(Int32)

Simple [✓ Incompatible] All

Cancel OK

The last step is to catch the second event. This one return an integer, thus we add to convert it into a string value before display it into the label. First, we add the link between the coolbean component and the value formatter. The source is “NombreAleatoire” and the destination is “Format”.



Create Link

Source:
coolbeans coolbeans1

MethodEvent
NombreAleatoire
NombresAleatoires
PropertyChanged

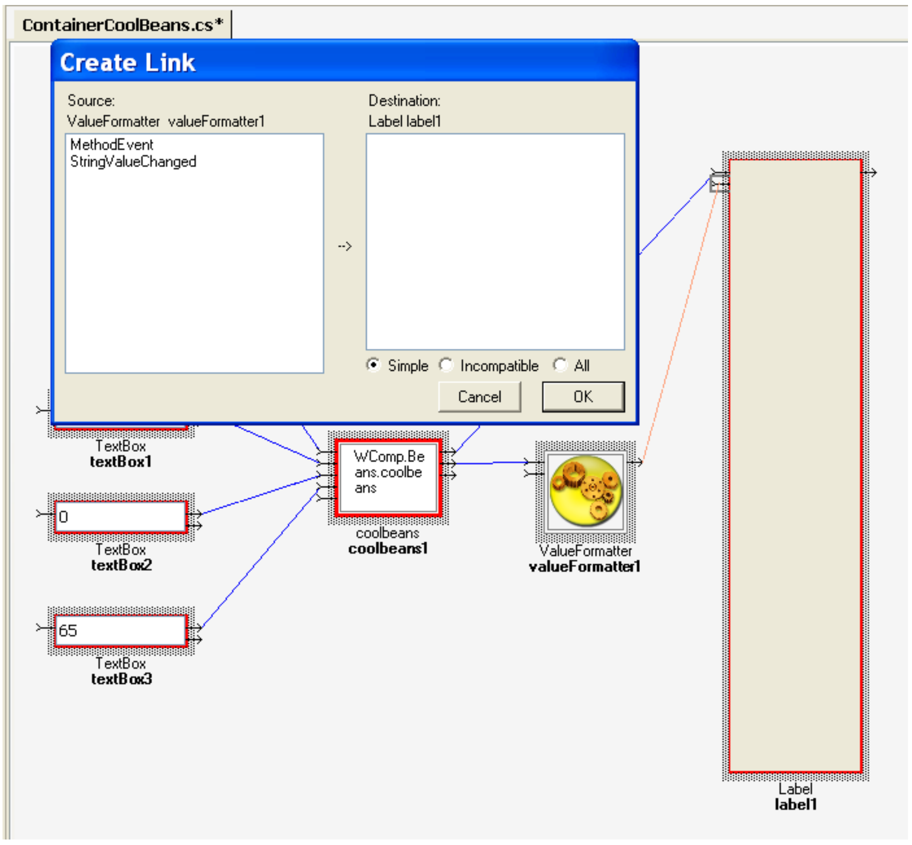
Destination:
ValueFormatter valueFormatter1

Void Format(Int32)
Int32 GetHashCode()
System.Object GetLifetimeService()
System.Type GetType()
System.Object InitializeLifetimeServic
Void set_Height(Int32)
Void set_Left(Int32)
Void set_Top(Int32)
Void set_Width(Int32)

[✓ Simple] Incompatible All

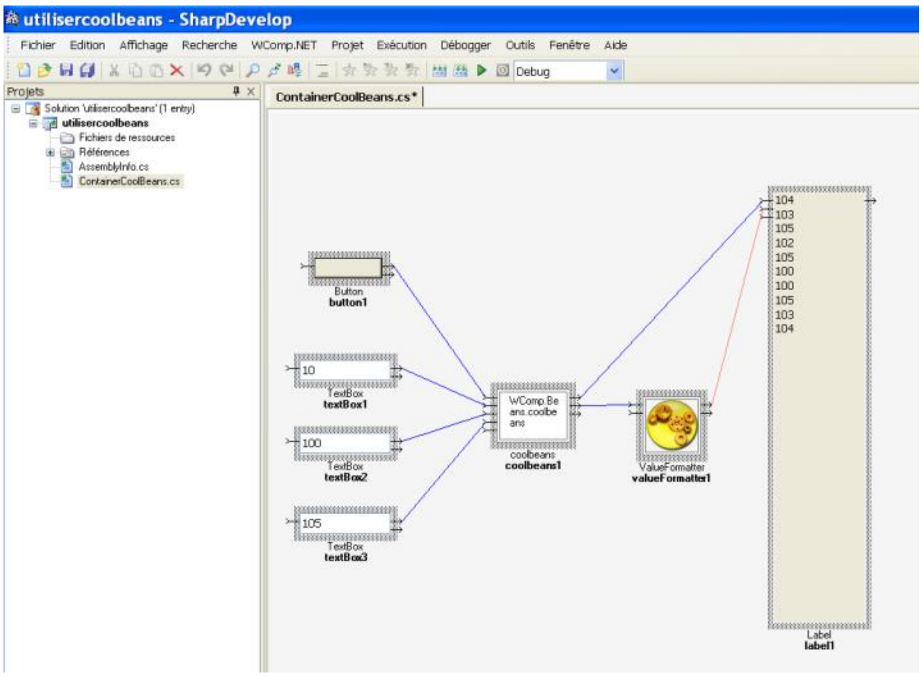
Cancel OK

Finally, we add the last link between the value formatter and the label. The source is “StringValueChanged” and the destination is “set_Text”.



2.3.6 SIXTH STEP : USE THE CONTAINER

Select some values into textboxes and click on the button a number or many appears into the label. Here is an example.



3 Conclusion

WComp is a powerful tool. With it is quite easy to make some beans and to use them. The only thing is to clearly understand the event programming concept and how to implement it.