# Les nouveautés de Windows Workflow Foundation 4.5

**Article par Jérémy Jeanson (MVP Connected System Developer)**

Développant avec .net depuis 2001, j'ai vécu les différentes évolutions du Framework, des premières Beta de la version 1.0 à aujourd'hui. Je surfe donc sur la vague .net depuis ses débuts, et j'adore cela.

À force de pratiques et d'expériences personnelles, je me suis spécialisé dans les architectures n-tiers exploitant Windows Communication Foundation, Windows Workflow Foundation et plus généralement l'interopérabilité des services.

## Sommaire de l'article

## Introduction

Avant de partir dans le vif du sujet, je souhaiterai attirer votre attention sur le fait qu'au moment où cet article est écrit, Windows Workflow Foundation 4.5 est en version Beta avec Visual Studio 11 (11.0.502114.1). Il est donc possible que la version finale diffère légèrement de ce qui est présenté ici.

## Amélioration générale du designer

La première chose qui frappera l'utilisateur habitué à Windows Workflow Foundation sera son designer. N'attendez pas un changement esthétique, mais à des performances qui surpassent largement Visual Studio 2010.

Avec la version Beta, on peut déjà observer des améliorations non négligeables de la réactivité du Designer pour :

- L'ouverture d'un workflow.
- L'affichage de la boite à outils.
- L'éditeur d'expression.
- Le sélecteur de types de la liste des variables et arguments
- … et bien d'autres…

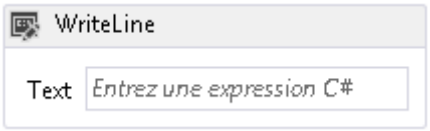
Personnellement, lors de la création de mon premier projet WF4.5 j'ai été surpris par la vitesse avec laquelle le Designer s'est ouvert. Le changement annoncé est donc bien présent.

## Associer des activités

Côté connexion entre les activités, autoassociation… etc… WF4.5 emploi les mêmes mécanismes que WF4 après la Plateform Update 1. La nouveauté se situera plutôt du côté des conteneurs.

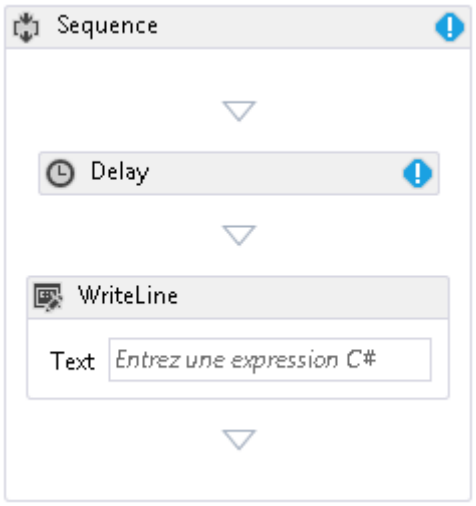
Avec WF4, si votre conteneur n'accepte qu'une activité, il est impossible d'ajouter une seconde activité avant ou après celle-ci (c'est le cas par exemple pur un workflow vide). Avec WF4.5, si vous tentez de glisser une activité, le designer va automatiquement ajouter une séquence pour englober l'activité déjà présente et la nouvelle.

Voici un exemple concret d'utilisation :
1 - Mon workflow contient une activité de Type « WriteLine ».




2 - Je tente ensuite de glisser une activité de type « Assign » devant ma première activité.

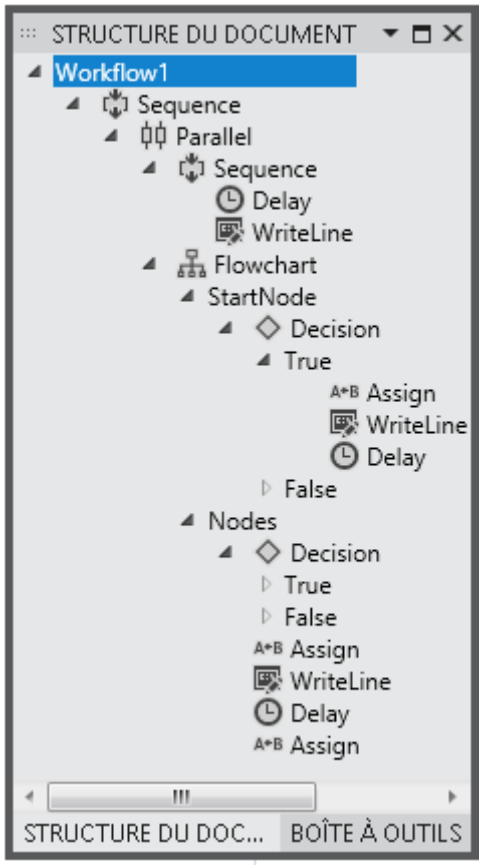
3 – Je relâche mon activité. Le designer l'a ajouté, et a englobé mes deux activités dans une séquence.




## La navigation

Le Designer de workflows se voit adjoindre une vue « Structure du document » . Cette vue représente le workflow sous la forme d'une arborescence que l'on peut parcourir à volonté. Plus le workflow est grand, et plus cette vue facilite la navigation.

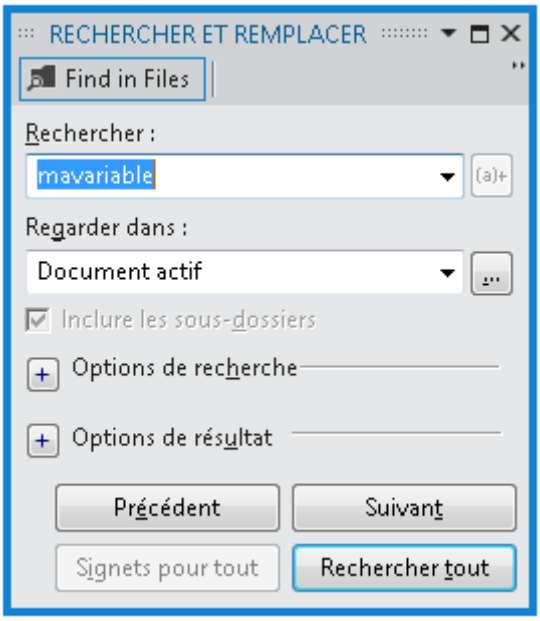
Cet outil est utilisable sur l'ensemble des type de workflows, du séquentiel à la machine à états.



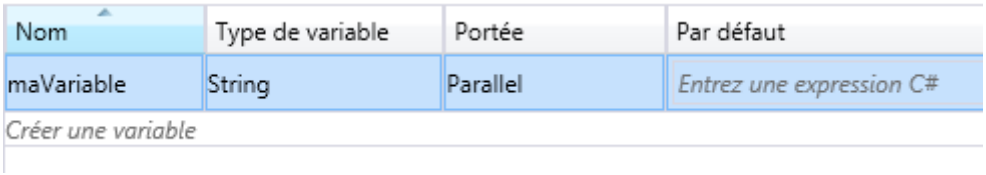
## La recherche

Fonctionnalité indispensable, mais totalement absente de WF4, il sera enfin possible d'utiliser la recherche de Visual Studio pour retrouver une variable ou un texte présent dans le design d'un workflow. C'est une avancée qui n'a pas de prix.

Ceci passe par l'utilisation de la fonctionnalité de recherche de Visual Studio (Ctrl+F par défaut). Nul besoin d'utiliser un outil spécifique à WF.



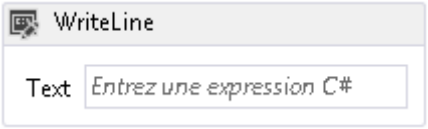

Exemple : Si une variable est trouvée, sont conteneur est immédiatement ouvert et la variable est sélectionnée :




## Les expressions

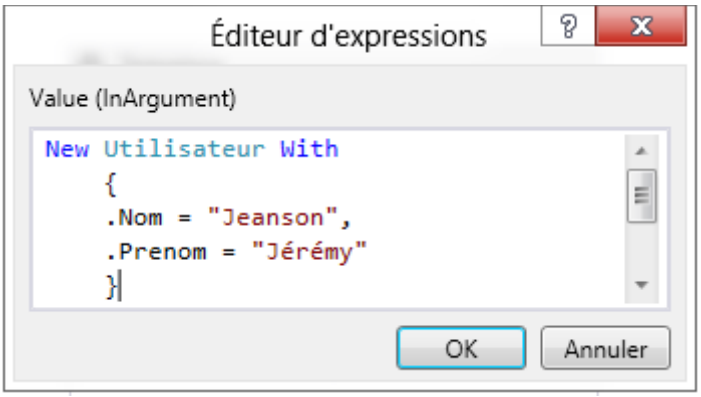
Jusqu'ici, les développeurs C# étaient obligés d'utiliser une syntaxe Vb pour édition des expressions .net présentes dans leurs workflows. Bonne nouvelle, avec WF4.5 il est maintenant possible d'écrire des expressions en C#.

Pour un projet C#, le langage d'expression sera C# et pour un projet Vb, le langage d'expression sera Vb. C'en est donc fini du monopole de Visual Basic.



Par exemple, voici une expression C# que j'ai utilisée pour instancier un objet :


Pour un projet Vb on aura l'équivalent suivant :




Ceci réjouira certainement les nouveaux utilisateurs de WF. Mais, cela risque aussi d'inquiéter les personnes désireuses de migrer leurs workflows de .net 4 ver .net 4.5. Pas de panique, la migration est indolore. Par défaut, Visual Studio conservera le langage d'expression que vous utilisiez. Les expressions ne seront donc pas modifiées.

Concrètement : les fichiers Xaml et Xaml vont maintenant contenir une propriété permettant de spécifier le langage utiliser (ligne mise en évidence sur la capture d'écran suivante) :

En fonction des cas, celle-ci se trouve directement sur la racine du workflow :

```
<Activity x:Class="Demos.WF45.Expressions.Activity1"
          xmlns="http://schemas.microsoft.com/netfx/2009/xaml/activities"
          mc:Ignorable="sap2010"
          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
          sap2010:ExpressionActivityEditor.ExpressionActivityEditor="C#"
          xmlns:sap2010="http://schemas.microsoft.com/netfx/2010/xaml/activities/prese
          xmlns:sco="clr-namespace:System.Collections.ObjectModel;assembly=mscorlib"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
```

Ou dans le contenu de la racine

```
<Activity mc:Ignorable="sap sap2010 sads" x:Class="Demos.WF45.Expressions.Workflow1"
 xmlns="http://schemas.microsoft.com/netfx/2009/xaml/activities"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:sads="http://schemas.microsoft.com/netfx/2010/xaml/activities/debugger"
 xmlns:sap="http://schemas.microsoft.com/netfx/2009/xaml/activities/presentation"
 xmlns:sap2010="http://schemas.microsoft.com/netfx/2010/xaml/activities/presentation"
 xmlns:scg="clr-namespace:System.Collections.Generic;assembly=mscorlib"
 xmlns:sco="clr-namespace:System.Collections.ObjectModel;assembly=mscorlib"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <x:Members>
    <x:Property Name="argument1" Type="InArgument(x:String)" />
  </x:Members>
  <sap2010:ExpressionActivityEditor.ExpressionActivityEditor>C#</sap2010:ExpressionAct
  <sap2010:WorkflowViewState.IdRef>Demos.WF45.Expressions.Workflow1_1</sap2010:Workflc
```

Sur la capture suivante, on constate aussi l'arrivée d'une nouvelle structure pour les définitions de workflows. Les namespaces et les références ajoutées aux workflows sont maintenant dans des conteneurs distincts (NamespacesForimplementation et ReferenceForImplementation. Cette partie du Xaml devient enfin lisible et facilement éditable.

```
<Activity mc:Ignorable="sap sap2010 sads" x:Class="Demos.WF45.Expressions.Workflow1"
 xmlns="http://schemas.microsoft.com/netfx/2009/xaml/activities"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:sads="http://schemas.microsoft.com/netfx/2010/xaml/activities/debugger"
 xmlns:sap="http://schemas.microsoft.com/netfx/2009/xaml/activities/presentation"
 xmlns:sap2010="http://schemas.microsoft.com/netfx/2010/xaml/activities/presentation"
 xmlns:scg="clr-namespace:System.Collections.Generic;assembly=mscorlib"
 xmlns:sco="clr-namespace:System.Collections.ObjectModel;assembly=mscorlib"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <x:Members>
    <x:Property Name="argument1" Type="InArgument(x:String)" />
  </x:Members>
  <sap2010:ExpressionActivityEditor.ExpressionActivityEditor>C#</sap2010:ExpressionAct
  <sap2010:WorkflowViewState.IdRef>Demos.WF45.Expressions.Workflow1_1</sap2010:Workflc
  <TextExpression.NamespacesForImplementation>
    <sco:Collection x:TypeArguments="x:String">...</sco:Collection>
  </TextExpression.NamespacesForImplementation>
  <TextExpression.ReferencesForImplementation>
    <sco:Collection x:TypeArguments="AssemblyReferen">...</sco:Collection>
  </TextExpression.ReferencesForImplementation>
  <Sequence>
    <Assign sap2010:WorkflowViewState.IdRef="Assign_1" />
    <WriteLine sap2010:WorkflowViewState.IdRef="WriteLine_10" />
    <sap2010:WorkflowViewState.IdRef>Sequence_4</sap2010:WorkflowViewState.IdRef>
    <sads:DebugSymbol.Symbol>d3BDOlxVc2Vyc1xKZXJlbXlcRG9jdW1lbnRzXFZpc3VhbCBTdHVkaW8g
  </Sequence>
  <sap2010:WorkflowViewState.ViewStateManager>
    <sap2010:ViewStateManager>...</sap2010:ViewStateManager>
  </sap2010:WorkflowViewState.ViewStateManager>
</Activity>
```

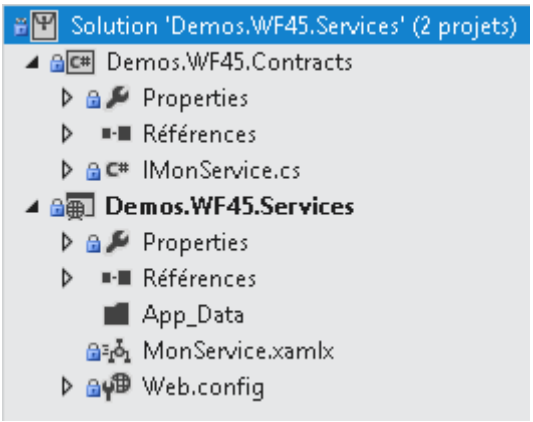
NOTE : Les éléments liés au positionnement, dimensionnement, et plus généralement tout ce qui est lié au design, fait maintenant partie d'une section spécifique « ViewStateManager ». On pourra donc envisager de supprimer rapidement ces contenus inutiles sans risque d'erreur, avant de publier un workflow.

Les éléments de débogage restant dans la définition du workflow.

## Workflows Services et contract First

Grande avancé pour les utilisateurs de services de workflows : les workflows pourront être construit en respect d'un contrat préalablement établi.

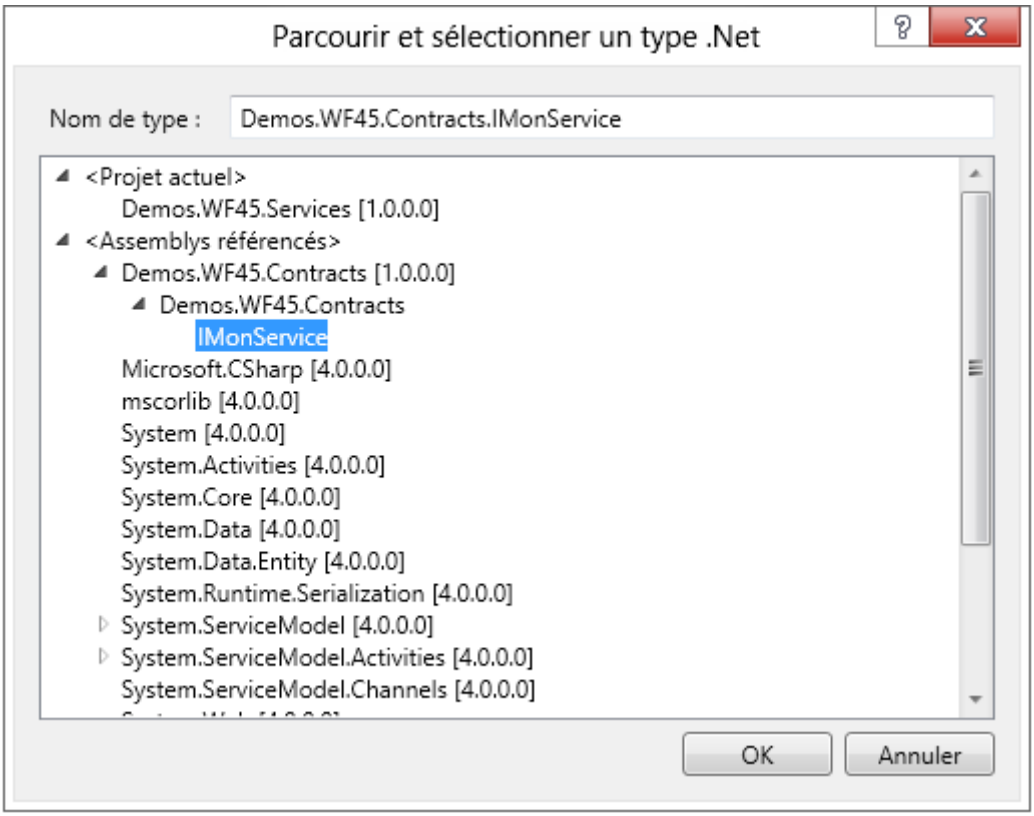
Voyons maintenant comment cela fonctionne. Pour commencer, je vais créer deux projets. Le premier contient mes contrats et le second mes services :



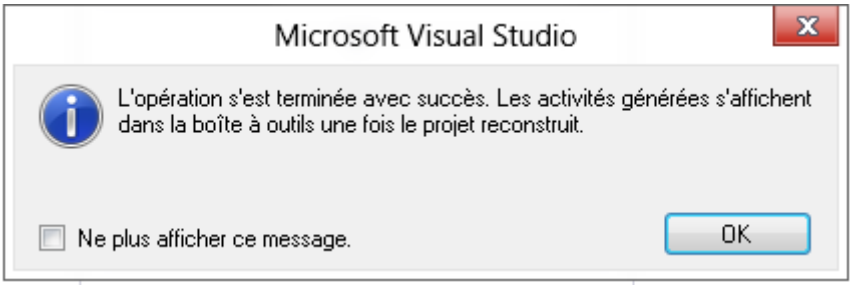

Dans le projet des contrats, j'ai créé le contrat suivant :

```
namespace Demos.WF45.Contracts
{
    [ServiceContract]
    public interface IMonService
    {
        [OperationContract]
        String SayHello(String prenom);
    }
}
```

Après compilation du projet contenant les contrats, on peut ajouter celui-ci comme référence au second projet (celui des services). Afin que WF prenne en compte les contacts, il est utile de demander d'importer de ceux-ci. Pour réaliser cette manipulation, il suffit de faire un click droit sur le projet des services et d'utiliser la commande « Importer le contrat de service ». Cette commande permet d'ouvrir une boite de dialogue nous demandant quel contrat doit être importé :

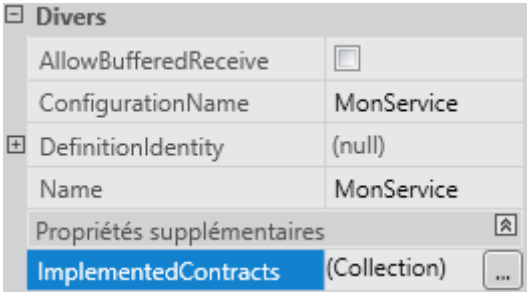



Après validation, une seconde boite de dialogue s'ouvre afin de nous indiquer que des activités ont été générées.

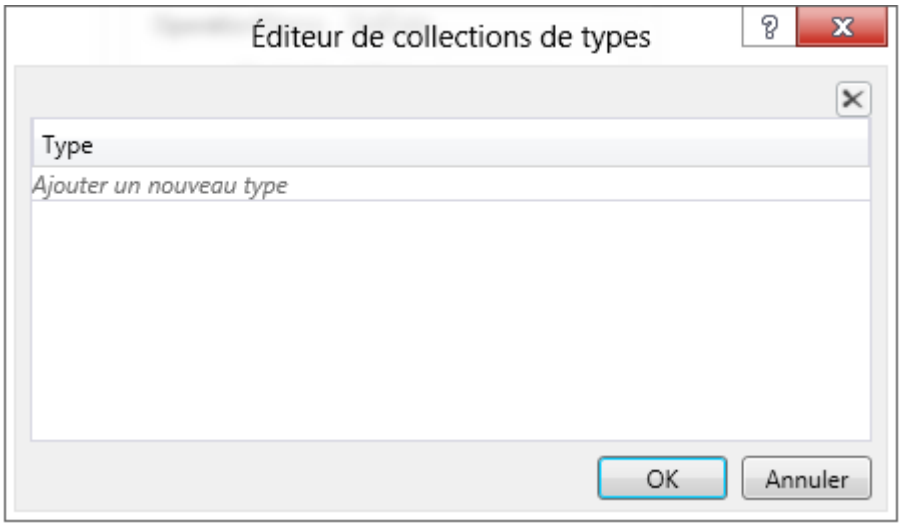



Les activités ainsi générées pourront être ajoutées à notre workflow afin que celui-ci respecte le contrat.

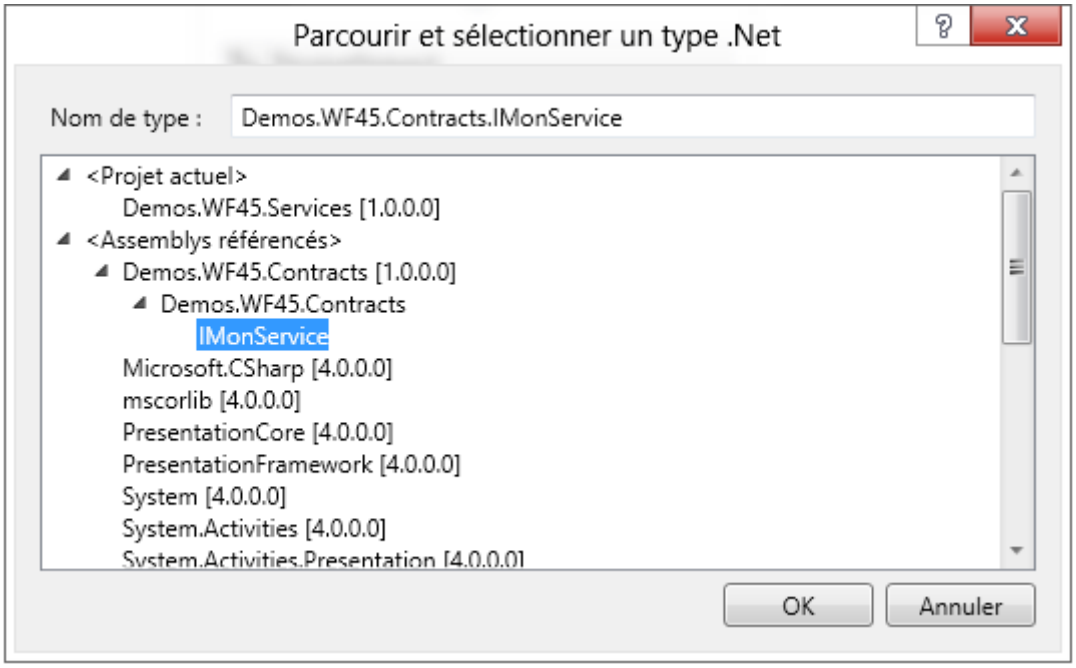
Côté service, il reste maintenant à indiquer au workflow qu'il doit respecter le contrat. Pour ceci, il faut ouvrir le workflow (fichier .xamlx) et afficher ses propriétés.



Avec WF4.5, la liste a été revue et permet maintenant de spécifier une liste de contracts : « IplementedContracts ». À l'ouverture, l'éditeur de cette propriété affiche une liste vide telle que ceci :

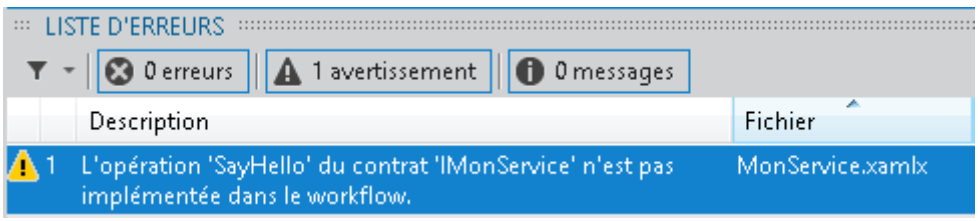



Un click sur « Ajouter un nouveau type » permet d'avoir une liste déroulante dans laquelle on choisira « Rechercher des types » afin de sélectionner notre contrat :



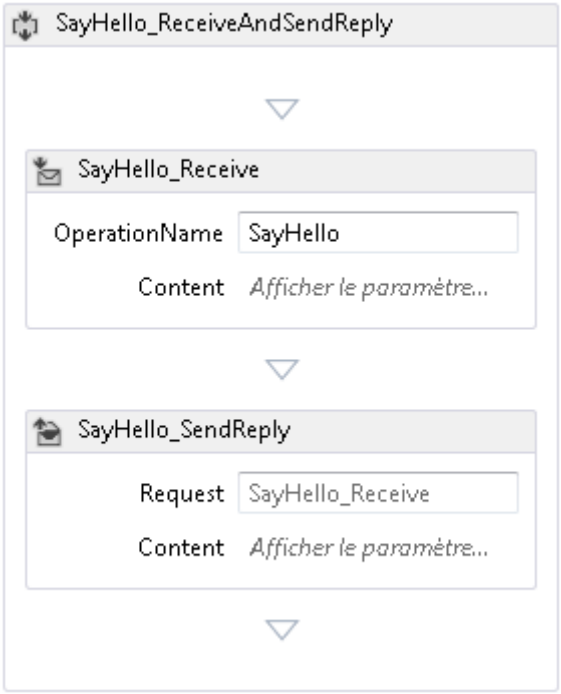

On choisit son contrat et ensuite on valide la collection de contrats.

A ce stade, un warning est visible dans la vue des erreurs et warnings :



Ceci est normal, car nous n'avons pas encore glissé dans notre workflow les activités générées par Visual Studio :


Il suffit alors de glisser les activités du contrat dans le workflow (dans notre cas, il n'y en a qu'une seule).




Si on ouvre les paramètres des messages, on verra qu'ils respectent les arguments et le type de retour de la méthode SayHello du contrat. Le « contract first » devient donc réalité avec WF4.5, et son implémentation est des plus accessibles.

Note : Il faudra bien entendu penser à passer la propriété « CanCreatInstance » de l'activité SayHello_Receive

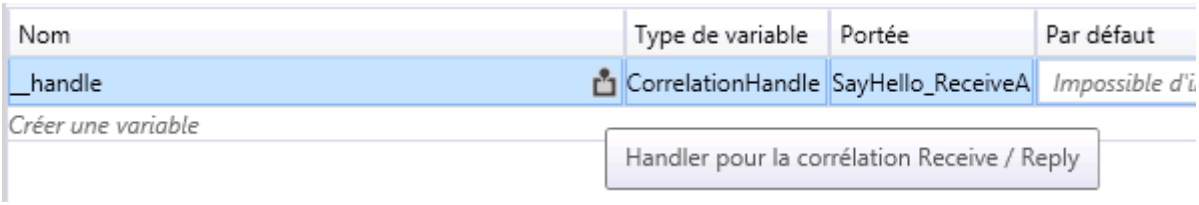
## Annotations

Il sera enfin possible d'ajouter des commentaires dans le designer de workflow. On peut annoter :

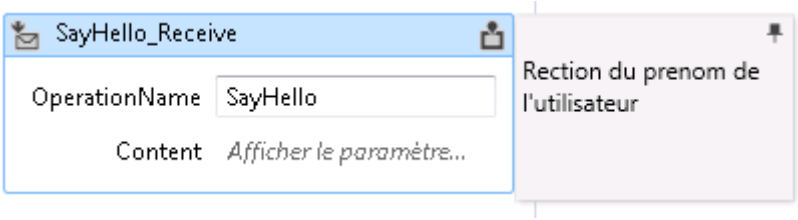
- Les activités.
- Les variables.
- Les arguments.

Chacun de ces éléments se voit adjoindre un menu contextuel permettant l'ajout, l'édition et la suppression des annotations.

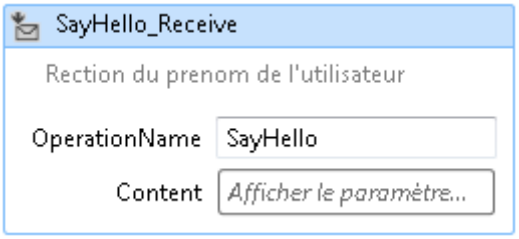
Pour une variable ou un argument, la note est représentée par un petit pictogramme qui fait apparaitre son contenu quand on le survol.




Pour une activité, il existe deux modes d'affichages. Par défaut, le fonctionnement est similaire à l'annotation de variables.




Si on utilise la punaise présente sur l'annotation, il est possible de la voir afficher en permanence. L'annotation apparait alors entre le titre de l'activité et son contenu. Si vous avez personnalisé les templates utilisés par vos DesignerActivity il faudra donc veillez à ce que ceux-ci soient compatibles avec les annotations.




Petit raffinement supplémentaire, trois commandes ont été ajoutées aux activités :

- Afficher toutes les Annotations.
- Masquer toutes les Annotations.
- Supprimer toutes les Annotations.

Comme leurs noms l'indiquent, il est donc possible en une seule action, d'afficher, masquer ou supprimer l'ensemble des annotations. À ce niveau donc, les choses n'ont pas été faites à moitié. J'apprécie tout particulièrement la commande Supprimer qui sera bien pratique pour supprimer des notes « trop personnelles » avant de distribuer un workflow à un tiers.

## La compilation

Sur le plan de la compilation, il y aura effectivement quelques améliorations liées comme toujours à l'amélioration de Visual Studio et de son compilateur. Côté spécificités WF, il y aura une grande nouveauté : des erreurs et des warnings seront affichés à la compilation.

Cette information peut sembler étrange, mais il faut savoir qu'avec WF4, ces informations ne sont accessibles que si les définitions de workflows sont ouvertes. Avec WF4.5, nous pourrons donc être informés des éventuelles erreurs et warnings quels que soient les fichiers ouverts.

## Le versioning

Le versioning est un problème qui se pose avec des workflows, à partir du moment que l'on a une ou plusieurs instances de workflows qui sont en cours d'exécution (ou qui sont persistées) et que l'on met veut mettre à jour la définition du workflow. On se trouve alors fasse à la plusieurs difficultés :

- Quelle définition va utiliser une instance qui doit être ravivée ?
- Comment savoir, quelle définition du workflow est utilisée par telle ou telle instance ?
- Comment mettre à jour les instances persistées avant qu'elles n'aient besoin d'être ravivées ?
- … etc…

Avec WF4, il n'existe pas de solution « out of the box ». Chacun doit trouver la solution qui répondra au mieux à sa situation.

Avec WF4.5, les choses vont changer radicalement. La réponse de WF4.5 à la problématique du versioning est basée sur trois axes :

- WorkflowIdentity : chaque instance de workflow pourra se voir adjoindre des informations liées à son identité (nom et version de la définition de workflow utilisée) qui permettra d'identifier quelle version du workflow utiliser.
- WorkflowSericeHost : cet hôte sera équipé d'un mécanisme interne lui permettant d'identifier automatiquement les WorkflowIdentity d'un service xamlx. Il servira donc automatiquement les bonnes versions de workflows à ces clients.
- Dynamic Update : il s'agit d'un mécanisme en deux étapes. Dans un premier temps, il permet d'établir une cartographie des différences existantes entre une ancienne et une nouvelle version d'un même workflow. Dans un second temps, on pourra utiliser cette cartographie sur des instances persistées afin de les mettre à jour, pour qu'elles deviennent compatibles avec la nouvelle version. Après application, les instances persistées utiliseront donc la nouvelle version du workflow.

Ces fonctions étant actuellement en cours de réalisation, nous ne pouvons malheureusement présenter que très peu d'exemples de code.

On connait cependant la manière dont on pourra gérer les WorkflowIdentity. Par exemple, pour associé une identité à une instance de workflow, on pourra écrire ceci :

```
// Création d'un identité
WorkflowIdentity identity = new WorkflowIdentity(
    "Workflow1",                // Nom : exemple, non du workflow
    new Version(1, 0, 0, 0),    // Version
    "Demos.WF45.Activities");   // Package : exemple, assembly des activités

// Crétion d'une instance de workflow avec l'identité voulue
WorkflowApplication application = new WorkflowApplication(
    new Workflow1(),
    identity);

application.Run();
```

Pour restauré une instance de workflow persistée, on pourra utiliser des VersionMismatchException afin de vérifier que l'on utilise la bonne WorkflowIdentity :

```
try
{
    // Création d'un identité
    WorkflowIdentity identity = new WorkflowIdentity(
        "Workflow1",                // Nom : exemple, non du workflow
        new Version(1, 0, 1, 0),    // Version
        "Demos.WF45.Activities");   // Package : exemple, assembly des activités

    // Crétion d'une instance de workflow avec l'identité voulue
    WorkflowApplication application = new WorkflowApplication(
        new Workflow1(),
        identity);

    application.Load(id);
}
catch (VersionMismatchException ex)
{
    // ex.ExpectedVersion : Version de l'identité utilisé quand le workfow a persisté
    // ex.ActualVersion   : Version de l'identité utilisé dans le bloc try
}
```

Note : id est une variable de type Guid, utilisée pour l'exemple.

## Ce qu'il n'y aura pas dans WF4.5

WF4.5 a été bâti sur les retours des utilisateurs de WF4 et WF3.5. Malheureusement, l'une des fonctionnalités les plus demandées ne sera pas de la partie lors de la sortie en RTM de .net 4.5. Il s'agit de l'intégration du designer de workflows dans Silverlight.

Rien ne dit par contre que l'idée est totalement abandonnée ou qu'elle ne fera pas partie d'une mise à jour du Framework ; comme ce fut le cas pour la StateMachine de WF4 intégré suite à la Plateform Update 1 de .net 4.0.

## « Happy End »

Comme vous pouvez le constater, nous avons jusqu'ici, très peu parler de nouvelles activités. Ceci est volontaire. Pour la Beta, Microsoft s'est concentrée sur la partie fonctionnelle de Workflow Foundation et son Runtime. Il a cependant été annoncé qu'au vu du succès et des retours faits sur les packs d'activités mis à disposition sur CodePlex (http://wf.codeplex.com/), plusieurs d'entre elles devraient rejoindre le Framework .net 4.5.

Il ne nous reste plus qu'à attendre ;)