2013

Pratique : WComp Plateforme de Composition

S. Lavirotte, J.-Y. Tigli, V. Hourdin, G. Rey

Université de Nice – Sophia Antipolis

26/08/2013

TD
WComp 3.2 : Plateforme de Composition

1 Découverte de WComp

Le modèles présentés dans ce sujet (LCA et SLCA) ont été publiés dans les articles ci-dessous :

> **Jean-Yves Tigli, Stéphane Lavirotte, Gaëtan Rey, Vincent Hourdin et Michel Riveill.** « *Lightweight Service Oriented Architecture for Pervasive Computing* ». International Journal of Computer Science Issues (IJCSI), volume 4, numéro 1, pages 1-9, septembre 2009. ISSN 1694-0814
>
> **Nicolas Ferry, Vincent Hourdin, Stéphane Lavirotte, Gaëtan Rey, Jean-Yves Tigli et Michel Riveill**. « Models *at Runtime: Service for Device Composition and Adaptation* ». Dans Proceedings of the 4th International Workshop Models@run.time at Models 2009 (MRT'09), pages 51-60, Denver, Colorado, USA, octobre 2009. ACM / IEEE Computer Society.

Vous pouvez vous référer à la documentation technique de référence disponible en ligne à l'adresse suivante pour l'installation et la prise en main de l'environnement WComp :

https://download.wcomp.fr/Tutorial/Polytech/

Commencez par démarrer SharpWComp et créez un fichier WComp :

- Fichier / Nouveau / Fichier …
- WComp.Net / C# Container -> crée un nouveau fichier Container.cs (onglet en haut de la zone de travail)
- Pour pouvoir manipuler les composants il faut activer la représentation graphique du Container (onglet WComp.NET en bas de la zone de travail, passage sur cet onglet automatique en SharpWComp 3.2).

N'oubliez pas que vous pouvez sauvegarder vos assemblages de composants avec l'option Export du menu WComp.NET.

2 Composition Locale : Modèle LCA

Le modèle LCA (Lightweight Component Architecture) est un modèle de composants légers vous permettant de créer des assemblages de composants pour réaliser vos applications. Nous allons voir comment utiliser et créer des composants : composants fournis avec SharpWComp, composants proxy vers un service ou encore créer vos propres composants pour répondre à vos besoins.

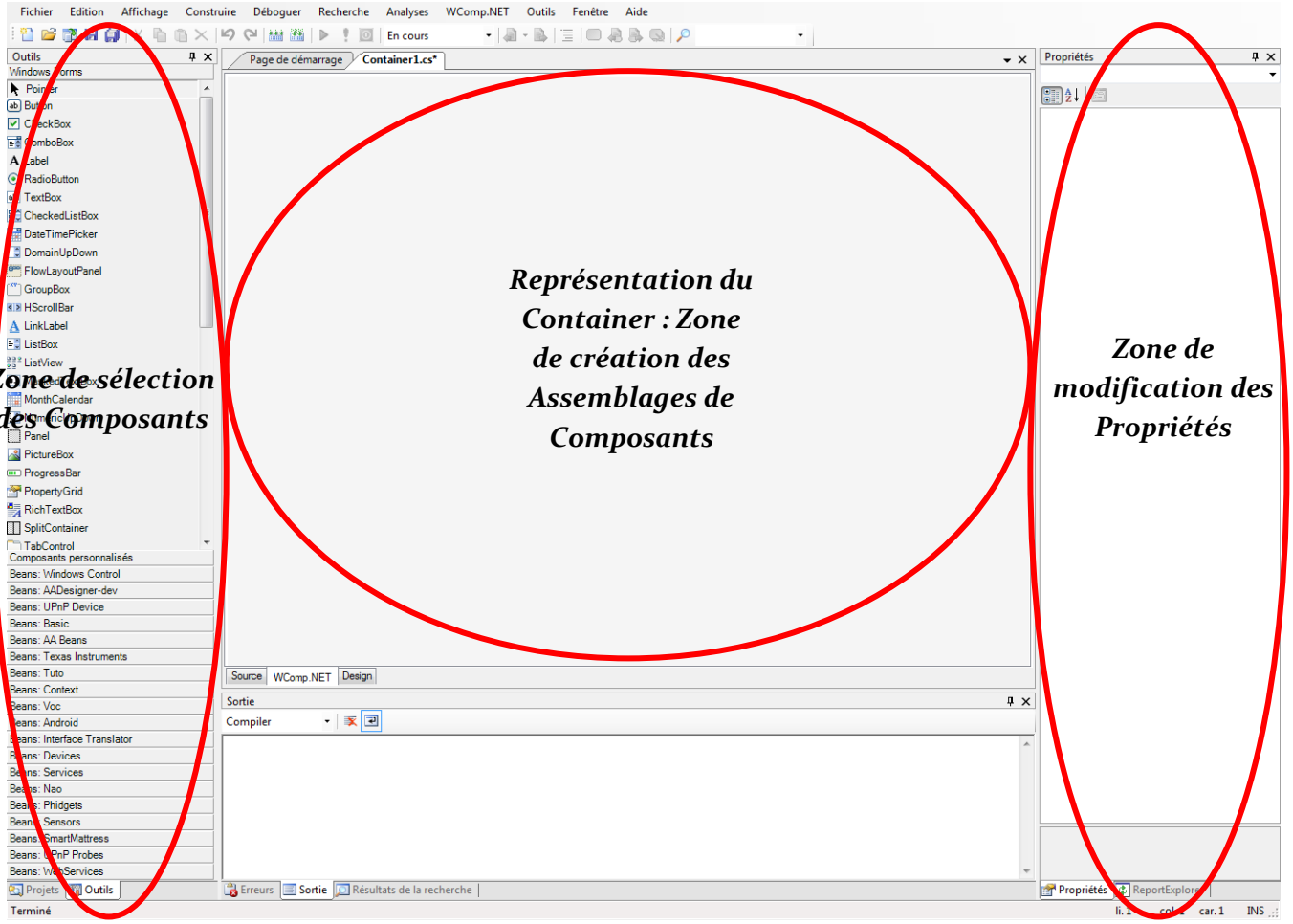
Nous verrons aussi dans cette section comment relier un événement émis par un composant à un appel de méthode d'un autre composant ce qui permet d'établir des communications entre les composants.

2.1 Mise en place d'une application par assemblage de composants

2.1.1 Présentation de l'interface de l'Addin WComp dans SharpDevelop

WComp est un Addin de SharpDevelop ; celui-ci s'intègre donc à l'environnement de développement. Voici ce que vous devez avoir sur votre écran après avoir créé un container comme décrit plus haut.

TD
WComp 3.2 : Plateforme de Composition



2.1.2 Propriétés

Les composants BeanWComp possèdent des propriétés qu'il est facile de modifier. Dans l'interface de SharpDevelop, quand vous sélectionnez un composant dans le container et ses propriétés s'affichent sur la droite. Créez un composant de type Button (onglet Windows Forms), et modifier la propriété correspondant au texte affiché dans le bouton (propriété Text). Affectez la valeur « Hello World ! » à la propriété Text de ce bouton.

2.1.3 Evènements simples

Un événement simple est défini par le fait que les données portées par l'événement sont compatibles avec la signature de la méthode appelée.

Une illustration très simple de cette connexion est par exemple de relier un Button avec un RadioButton (onglet Windows Forms). En reliant l'événement Click du Button que nous avons créé précédemment avec la méthode PerformClick (qui ne prend pas de paramètre), on a un événement simple. Il n'y a pas d'ajout ou de transformation de l'ordre des paramètres (la signature est compatible).

2.1.4 Evènements complexes

Lorsque la signature est incompatible (pas assez d'événements ou type des événements différents), on parle alors d'événement complexe.

Prenons l'exemple suivant. Si en cliquant sur le bouton, nous souhaitons envoyer la valeur du texte affiché sur ce bouton au composant suivant (une TextBox par exemple), l'événement Click n'envoyant aucune donnée, il faudra faire un rétro-appel sur le composant appelant pour compléter la signature de la méthode appelée.

Essayer de mettre en place cet exemple en reliant votre Button « Hello World » avec un composant TextBox. Vous devez sélectionnez une méthode incompatible (toutes les méthodes s'affichent alors) et sélectionnez la methode Set_Text de la TextBox. Vous aurez alors accès à l'ensemble de la liste des getters sur le composant d'origine pour tenter de trouver une méthode vous permettant de fournir le ou les paramètres (get_Text en l'occurence).

2.2 Création d'un Composant Bean

Il peut s'avérer nécessaire de créer vos propres composants si les composants existants ne correspondent pas à vos besoins. L'environnement offre la possibilité de créer un composant à partir d'un squelette de code. Nous allons donc créer un composant permettant de compter le nombre de click effectués sur le bouton.

2.2.1 Création et compilation d'un composant

Après avoir quitté et redémarré SharpWComp, vous pouvez créer une nouvelle solution pour la création du composant souhaité :

- Fichier / Nouveau / Solution …
- WComp.NET / WComp Bean Combine (nommer cette solution Compteur)

Ajoutez un fichier à votre nouvelle solution :

- Clic droit sur votre solution / Ajouter / Nouveau Fichier …
- WComp.Net / C# Bean -> crée un nouveau fichier Bean1.cs (que l'on peut renommer en Compteur.cs)

Il ne vous reste plus qu'à modifier le squelette de code pour donner à ce nouveau composant le comportement souhaité.

Nous allons faire un composant qui nous permettra de faire clignoter une des lumières du feu tricolore. Ce composant disposera tout d'abord d'une **propriété** `Valeur` permettant de stocker (ou de modifier) le nombre d'actions effectuées. Pour coder cette propriété, inspirez-vous du code contenu dans le squelette de code d'un composant Bean.

Ce composant aura une **méthode** `Increment` qui, à chaque fois que cette méthode est appelée, on incrémentera la valeur du compteur (Valeur) et on émettra un événement avec cette valeur. Donc la dernière partie de votre composant est donc de définir un **événement** `ValeurChangee` qui émettra la valeur du compteur.

Ecrire le code correspondant, le compiler et ajouter la librairie créée dans le dossier contenant les composants (dossier `C:\Program Files (x86)\SharpDevelop\3.0\Beans\`). Un nouveau composant Compteur est maintenant disponible pour vos assemblages. Si vous n'avez rien précisé dans la directive [Bean] vous trouverez votre composant dans la catégorie Beans : Basic. Si vous souhaiter créer une catégorie particulière pour y ranger ce nouveau composant, vous devrez spécifier comme directive en tête de votre classe :

```
[Bean(Category="Tuto")]
```

2.2.2 Créer un assemblage pour la mise en œuvre de l'application

Faite un assemblage qui permettra de compter le nombre de fois où on clique que le bouton. Vous aurez besoin du bouton, de votre composant, d'une TextBox où afficher le résultat et peut-être d'un composant supplémentaire (à vous de voir).

TD
WComp 3.2 : Plateforme de Composition

2.3 Prototypage rapide : Génération du code source et compilation de l'application

L'application que venez de construire dynamiquement vous satisfait pleinement et vous souhaitez en faire une application que vous voulez utiliser sans l'environnement WComp c'est possible. Grâce à la transformation de modèle, en basculant sur l'onglet source, il est possible de voir le code généré correspondant à votre assemblage de composants. Il vous suffit de créer une solution C#, d'y ajouter la classe que vous avez sauvegardé et qui correspond à votre assemblage de composant et de rajouter les références vers les dll des beans que vous utiliser. Vous pouvez alors compiler votre applications pour en faire un exécutable.

Si vous utilisez des widgets dans votre assemblage, vous pouvez aussi réaliser la mise en forme de votre interface graphique grâce à l'onglet Design.