2009

Pratique : Composition dans WComp

S. Lavirotte, J.-Y. Tigli, G. Rey, N. Ferry

Université de Nice – Sophia Antipolis

09/07/2009

TD
Composition dans WComp 2.0

1 Découverte de WComp

Vous pouvez vous référer à la documentation disponible en ligne ainsi qu'aux vidéos de démonstrations disponibles à l'adresse suivante pour l'installation et la prise en main de l'environnement WComp :

http://rainbow.i3s.unice. fr/wikiwcomp/doku.php?id=download_telechargement

Commencez par démarrer SharpWComp et créez un fichier WComp :

- Fichier / Nouveau / Fichier …
- WComp.Net / C# Container -> créé 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).

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

2 Composants proxy pour l'orchestration de Services

2.1 Intégration d'un Web Service pour Dispositif UPnP

Nous souhaitons pouvoir accéder dans WComp aux dispositifs de type UPnP. Pour cela nous allons devoir générer un composant proxy pour le service UPnP découvert. Prendre par exemple le dispositif UPnP Light que vous avez créé lors du TD sur UPnP :

- Fichier / Nouveau / Fichier …
- WComp.NET / UPnP Device WebService Proxy
- Sélectionner le dispositif Light dans la liste, ainsi que toutes les méthodes et variables d'état auxquelles vous souhaiterez avoir accès via ce composant proxy. Cliquer sur suivant puis sur Terminer. Vous venez de générer un composant proxy pour ce service UPnP.
- Recharger les composants pour avoir le nouveau composant généré (Menu WComp.NET / Reload Beans…)

Rechercher le composant dans la catégorie Beans : UPnP Device (onglet Outils) et l'instancier dans le container.

Pour tester ce composant et vérifier qu'il fonctionne bien, nous allons lui connecter des boutons pour le contrôler.

3 Modèle LCA

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

3.1.1 Evènements simples

Nous souhaitons pouvoir allumer et éteindre la lampe avec 2 buttons. Créer les 2 boutons : un bouton On et un bouton Off, et les relier au dispositif lampe afin d'appeler les méthodes `On()` et `Off()` du dispositif.

3.1.2 Evènements complexes

Avoir deux boutons pour le contrôle de la lampe peut ne pas s'avérer intéressant. Nous souhaitons maintenant pouvoir aussi contrôler la lampe avec une `CheckBox`.

Créer 1 `CheckBox` et la relier au composant proxy de la Lampe afin de piloter le dispositif à l'aide de la méthode `SetStatus(boolean)`.

3.2 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 propose la possibilité de créer un composant à partir d'un squelette de code. Nous allons donc créer un composant permettant de faire clignoter la lampe.

3.2.1 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 Blink)

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 blink.cs)

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

Nous allons faire un composant qui nous permettra de faire clignoter une lampe. Ce composant disposera tout d'abord d'une **propriété** `sleepProperty` permettant d'indiquer la fréquence d'émission de ces événements. Pour code cette propriété, inspirez vous du code contenu dans le squelette.
Ce composant aura une **méthode** `MethodBlink` qui lancera un thread (`ThreadLoop`) émettant à intervalle régulier et alternativement un événement booléen (vrai ou faux). Vous trouverez ci-dessous le corps des fonctions MethodBlink et ThreadLoop qu'il ne vous reste plus qu'à compléter.

```
        private Thread myThread;

        public void MethodBlink(bool on){
                if (on) {
                        myThread = new Thread(new ThreadStart(ThreadLoop));
                        myThread.Start();
                } else {
                        // TODO: on termine par l'envoi d'un événement à faux
                        //       (extinction de la lumière)
                        myThread.Abort();
                }
        }

        private void ThreadLoop(){
                while (Thread.CurrentThread.IsAlive) {
                        Thread.Sleep(sleepProperty);
                        // TODO: Faire alterner l'état de vrai à faux et vice versa
                        // TODO: Emettre la nouvelle valeur sous forme d'événement
                }
        }
```

Ecrire le code correspondant, le compiler et ajouter la librairie créée dans le dossier contenant les composants (dossier `C:\Program Files\SharpDevelop\Beans\`). Un nouveau composant Blink est maintenant disponible pour faire 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="Demo")]
```

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

Faire un assemblage pour tester votre composant à l'aide des dispositifs virtuels Switch et Lampe (dispositifs virtuels UPnP).

4 Modèle SLCA

Vous venez de découvrir et de manipuler l'interface graphique qui représente l'assemblage de composants dans un container. Nous allons maintenant étudier les services qui peuvent être associés au container afin de mieux comprendre l'architecture SLCA.

4.1 Interface de contrôle : Manipulation de l'assemblage par l'interface de Service du Container

Pour activer cette interface UPnP de votre container, vous devez activer l'option « Bind to UPnP Device » dans le menu WComp.NET. A l'aide de l'outil « Device Spy » d'Intel, découvrez les dispositifs associés au container WComp. Nous allons tout d'abord nous intéresser au dispositif `WCompNetAppli_Container1_cs_0` qui nous donne accès à l'interface de manipulation de l'assemblage. Ce service va permettre d'ajouter, de supprimer, des composants et des liens.
A l'aide de « Device Spy » abonnez-vous aux événements. Vous pourrez constater que des événements sont émis à chaque modification de l'assemblage (création/destruction de composant et de lien).
Testez les méthodes pour lister le contenu d'un container (composants et liens), et tenter d'instancier de nouveaux composants et lien et de les détruire à l'aide de l'interface fournie par le service comme par exemple un composant de type `System.Windows.Forms.Button`. Vous noterez que l'interface de création des composants permet de spécifier ou pas le nom de l'instance du composant.
A l'aide l'interface, créez un lien entre deux composants.

4.2 Interface fonctionnelle : Ajout de composants sondes et puits

Le deuxième dispositif (`WCompNetProbe_Container1_cs_0`) associé au container est une interface qui va nous permettre d'injecter des données depuis l'extérieur du service ou bien de récupérer des données à partir du service.

L'interface de ce dispositif est actuellement vide. La modification de cette interface est réalisée par l'ajout ou le retrait de composants sondes et puits. Ces composants sont dans la catégorie `Beans:Basic` de WComp : `EmitProbe` et `SourceProbe` qui vont respectivement permettre d'émettre des événements à partir de l'interface du service d'une part et d'émettre des événements à partir de données de l'assemblage.

4.2.1 Sonde permettant d'émettre une information à partir du container

Ajoutez un composant `SourceProbe` à votre assemblage. Vous pouvez constater que l'interface fonctionnelle du dispositif est automatiquement modifiée (ajout d'un service UPnP avec un événement `sourceProbe`).

Faire un assemblage avec une `TextBox` qui pour tout texte tapé dans la `TextBox` provoquera l'émission d'un événement pour chaque nouvelle valeur dans la zone de texte.

4.2.2 Sonde permettant d'injecter une information dans le container

Ajouter un composant `EmitProbe` à votre assemblage. Vous pouvez constater qu'un nouveau service est ajouté à l'interface fonctionnelle.

Faire un assemblage pour récupérer dans une `TextBox` qui permettra de récupérer la valeur de l'appel de méthode qui a été effectuée sur l'interface fonctionnelle du container.

5 Designers

Il est possible d'utiliser le modèle SLCA de WComp, et en particulier l'interface de contrôle, pour agir sur l'assemblage de composants en dehors de l'utilisation de l'interface graphique de SharpDevelop. Nous allons vous présenter deux outils permettant d'interagir avec l'assemblage de composants de WComp

5.1 Designer Textuel

Le TextDesigner permet de créer des composants et des liens en utilisant un langage de commandes. Voici quelques utiles :

- CheckBeans
- CheckLinks
- CreateBean <*BeanType*> <*instanceName*>

Vous retrouvez l'interface de contrôle du dispositif associé au container WComp, mais sous forme de commandes textuelles. Ceci vous permet donc de créer des scripts pour construire, détruire ou adapter des assemblages.

5.2 Designer UPnP

Le Designer UPnP va permettre, quant à lui, d'automatiquement générer le composant proxy et l'instancier dans le container auquel il est associé à chaque nouveau dispositif UPnP.

Pour le tester, après avoir lancer WComp et le Designer UPnP, activez la connexion (Bind to UPnP Device sur WComp) et lancez un nouveau dispositif UPnP (votre lampe ou interrupteur réalisé précédemment). Automatiquement, le code du composant proxy est généré et un composant de ce type est instancier et configuré avec les bonnes propriétés (adresse du serveur UPnP, etc.).