Inversion de controle en .Net, pattern, interet et outils

par Philippe Vialatte (philippe.developpez.com)

Date de publication : 02 June 2009

Dernière mise à jour :

Dans mon article precedent sur les principes SOLID, j'ai mentionne, brievement, le pattern Inversion de contrôle, qui est une forme plus générale du principe d'inversion des dépendances.

Comme ce pattern semble relativement peu connu (et encore moins utilise) dans la communaute .net francophone, cet article va s'attarder plus longuement sur les avantages de ce pattern, ainsi que sur les moyens de le mettre en oeuvre.

Ce pattern à été décrit dans

Presentation du Pattern "inversion de contrôle"

Le pattern Inversion de contrôle est un pattern d'architecture (certains diront un principe de conception plus qu'un pattern a proprement parler) dont le but est de diminuer le couplage entre deux modules en deplacant la resolutions des dependances en dehors du module appelant.

En programmation orientée objet, de facon classique, un object (classe ou module) va contenir un ensemble de dépendances sur d'autres objets, auxquels il va déporter tout ou partie de ses traitements. Le bon coté de la chose est que l'on évite que nos objets contiennent trop de comportements (les rendants difficiles à maintenir). Le mauvais coté est que chacun de ces objets référencé devient une dépendance forte, car notre objet appelant doit connaitre chacun des objets qu'il va utiliser avant de les instancier.

Par exemple, prenons le cas d'une classe d'accès aux données. L'exemple standard d'appel à une fonction située dans une classe d'accès aux données aura la forme suivante :

Si on utilise l'inversion de contrôle, on va représenter le même code de cette façon :

La différence est que la résolution de la dépendance est déportée en dehors du code "métier".

En d'autres termes, on va, dans un premier temps, definir un jeu d'interfaces de facon a ce que nos differents modules puissent communiquer par un contrat. Dans un second temps, on va "injecter" dans notre objet un autre objet répondant au contrat défini.

Utiliser l'inversion de controle offre les avantages suivants :

- Chaque systeme ne se concentre que sur sa ou ses taches principales.
- Les differents systemes ne font pas dhypothèses sur le comportement des autres systemes.
- Remplacer un systeme ne produit pas d'effets de bord sur les autres systemes, tant que le contrat d'origine est respecte.
- Dans le cas d'une nouvelle version d'un composant (ou d'un composant alternatif, comme un changement de framework de log, par exemple), il est plus facile de changer le composant appelé.

Types d'injection possibles

De façon classique, on va distinguer trois types d'injection. Ces possibilités sont offertes par tous les langages objets.

Injection par une propriete

A-1 - Mise en place

A-2 - Avantages & Inconvenients

Injection par une interface

B-1 - Mise en place

B-2 - Avantages & Inconvenients

Injection par le constructeur

C-1 - Mise en place

Le système d'injection le plus couramment utilisé est de passer la dépendance au constructeur. Dans notre exemple précédemment montré, cela donnera le code suivant :

C-2 - Avantages & Inconvenients

Conteneurs IoC disponibles en .net

Spring.Net

Spring.Net a l'avantage, sur les autres conteneurs, de la maturite. C'est en effet un portage en .net de Spring en Java.

Unity

Unity est le contreneur d'IoC propose par Microsoft. Quoique beaucoup plus récent que Spring, il est pratiquement aussi puissant...Et aussi complexe.

Ninject

Ninject est un des derniers nés des frameworks d'IoC. Sa particularité est d'être plus rapide, mais aussi beaucoup moins complexe que les autres. Ninject ne se base pas sur les fichiers XML comme Spring et Unity, mais exclusivemenjt sur des expressions lambda. Le revers de la médaille est qu'il ne peut pas être utilisé avec des versions du framework antérieures à la 3.5.

StructureMap

Comme d'habitude, je préfère garder le meilleur pour la fin ;).

StructureMap est un conteneur open source, dont le développement est encours depuis quatre ans (soit le plus vieux framework en dehors de Spring). Contrairement à Spring, il ne 'agit pas d'un portage d'un projet Java existant, mais bien d'un framework pensé et développé pour l'inversion de Controle en .Net.

Les dernières versions de StructureMap incluent la notation lambda que l'on retrouve aussi dans Ninject, mais supporte également l'utilisation de fichiers de configuration, et l'utilisation d'attributs de classe.

Le framework CommonServiceLocator

On à vu qu'utiliser un framework d'IoC permets de réduire les dépendances entre les différents composants de notre application, pour en augmenter la maintenabilité. Or, si on pousse la logique à l'extrême, il nos resetera toujours une dépendance, quoi que l'on fasse...sur Ike framework d'IoC.

Le projet CommonServiceLocator, disponible sur Codeplex, est justement la réponse à cette problèmatique. Il permets en effet d'utiliser une interface de plus haut niveau, et d'agir comme un pattern Adapter pour rerouter les appels au Service Locator au framework d'IoC choisi.

Conclusion

Remerciements