

Livre blanc

ADIBA

Des outils Microsoft pour industrialiser vos développements

Auteur : Sophiane SOUANE
Version 1.0

ADIBA

FACILITATEUR D'ARCHITECTURE

Synopsis

Ce livre blanc s'adresse aux chefs de projets, architectes et décideurs qui souhaitent initier une démarche d'industrialisation des développements sur la base des technologies Microsoft .NET.

Microsoft via l'équipe Patterns and Practices fournit un ensemble de recommandations et « meilleurs pratiques » automatisées au travers de « software factories ». C'est sur cette base que ce livre blanc introduit ces technologies et les frameworks sur lesquels ils reposent.

Se tenir au courant des différentes initiatives autour de l'industrialisation logicielle dans l'environnement Microsoft nécessite une veille constante. L'objectif de ce document est de fournir une vue globale sur les principaux outils proposés par Microsoft pour industrialiser vos développements.

Avertissement :

Les informations présentées dans ce document sont le résultat d'investigation et de synthèse de différents éléments issus d'expériences professionnelles et de sites d'informations sur les technologies dans le monde entier.

Les informations fournies n'engagent en aucune façon ni Adiba ni Microsoft.

Table des matières

1. Pourquoi industrialiser ses développements ?.....	4
2. Software Factories	5
3 GAT / GAX.....	8
4 DSL, le designer de designer	9
5 VSIP / Add-In / Macro	10
6 Retours d'expérience.....	11
7 Conclusion	12
8 Table de références	13

1. Pourquoi industrialiser ses développements ?

Les architectes Jack Greenfield et Keith Short de l'équipe Visual Studio Team System chez Microsoft Corporation tirent la sonnette d'alarme dans l'article [« Moving to Software Factories »](#) au sujet de la demande en développement logiciel qui croît régulièrement. La solution ne peut se cantonner à augmenter le nombre de développeurs, il s'agit dès lors d'améliorer la productivité de ces derniers en mettant à disposition des outils et des méthodes adéquats. De plus, les coûts de réalisation ne sont pas maîtrisés et les coûts de maintenance sont importants.

Sur la base de ce constat, la fabrique logicielle (ou Software Factory) apporte une réponse adéquate. Elle permet d'accélérer le processus de création de logiciels à l'aide de modèles et pratiques recommandées en générant un socle structurant. Le développeur pourra alors dans ce cadre assurer une implémentation de qualité.

Nous allons voir quels sont les outils Microsoft à notre disposition pour concrétiser au quotidien cette démarche d'industrialisation du cycle de vie d'un projet logiciel.

2. Software Factories

Les « software factories » mentionnées dans ce document sont les fabriques logicielles développées par l'équipe Patterns and Practices de Microsoft.

Ces outils de développement réunissent les meilleures pratiques en architecture logicielle, autorisent de la génération de code à partir d'une collection de composants techniques (frameworks, socles techniques, bibliothèques de composants) et proposent de la documentation mais aussi des implémentations de référence.

Je vous invite à consulter les liens dans la table des références pour aller plus loin dans l'exploration de ces fabriques logicielles.

2.1 Web Service Software Factory

2.1.1 Concept

WSSF est une fabrique logicielle facilitant le développement de services Web ASMX ou WCF (Windows Communication Foundation).

2.1.2 Vue globale

Cette fabrique propose des assistants permettant la génération de composants basés sur les messages habituellement utilisés pour l'interopérabilité.

Vous pouvez alors générer différents composants tel que la couche d'accès aux données, couche métier et couche de service WCF.

En déclarant votre base de données, vous pouvez générer des procédures stockées en base de données et la couche d'accès aux données associée.

Ensuite, vous pouvez générer une liste d'entités métiers et implémenter votre logique métier.

Enfin, vous pouvez exposer votre métier sous la forme de services WCF par le biais d'artefacts (Data Contracts, Fault Contracts, Service Contracts, Message Contracts et Service Implementation). WSSF se charge de normaliser le code WCF que vous générez.

2.2 Smart Client Software Factory

2.2.1 Concept

SCSF est une fabrique logicielle facilitant la construction d'applications composites Windows reposant sur le framework Composite UI Application Block (CAB).

Le CAB est un framework destiné à construire des applications clients riches composites sur la base de modules indépendants qui peuvent interagir.

Ce framework est utilisé pour les applications qui possèdent de nombreux écrans et qui nécessitent une souplesse dans leur développement et leur déploiement, notamment en termes de personnalisation.

2.2.2 Vue globale

SCSF propose des assistants permettant la génération du squelette du code de l'application qui chargera dynamiquement les modules développés.

En automatisant ces tâches, cette fabrique masque la complexité du framework CAB. En effet, vous pouvez simplement ajouter un nouveau module, créer une nouvelle vue de présentation, implémenter vos événements basé sur le mécanisme de l'EventBroker (design pattern « observateur » ou publication / abonnement qui permet la communication inter-modules à base de messages), etc...

L'applicatif, l'affichage et l'accès aux données sont séparées dans chaque module sur la base du design pattern dérivé de MVC (Model-View-Controller) : MVP (Model-View-Presenter).

Vous pouvez voir une vidéo de démonstration de SCSF sur le site www.adiba.fr.

2.3 Web Client Software Factory

2.3.1 Concept

WCSF est une fabrique logicielle facilitant la construction d'applications composites Web reposant sur le framework Composite Web Application Block (CWAB) et le Page Flow Application Block (PFAB).

CWAB est un framework destiné à construire des applications Web modulaires avec ASP.NET.

PFAB est un framework destiné à intégrer Windows Workflow Foundation dans une application Web pour gérer la navigation inter-pages dans le site.

2.3.2 Vue globale

WCSF propose ainsi de découper votre application Web en plusieurs modules compilables de manière indépendante et rattachés à une racine unique de la même manière que la fabrique logicielle SCSF que nous avons vu précédemment.

Une classe « Controller » est chargée de contrôler la navigation parmi les pages. Elle implémente le modèle Application Controller.

Toujours comme les autres fabriques logicielles, des assistants visuels sont proposés pour vous faciliter la conception et la génération de code pour votre application.

2.4 Mobile Client Software Factory

2.4.1 Concept

MCSF est une fabrique logicielle facilitant la construction de clients mobiles reposant sur les frameworks Composite UI Application Block (CAB) et Data Access Application Block portés pour le .NET Compact Framework.

2.4.2 Vue globale

De la même manière que les autres fabriques logicielles de clients Web ou Windows, MCSF fournit de la documentation d'architecture, une implémentation de référence et des assistants visuels automatisant le développement avec l'environnement Visual Studio.

Le découpage proposé contient 3 couches : couche de présentation, couche métier et couche d'accès aux données.

3 GAT / GAX

Guidance Automation Toolkit est une extension à Visual Studio permettant d'élaborer des modèles dans le but d'automatiser la génération de code. C'est le successeur des « Enterprise Template Projects » (ETP) avec Visual Studio 2003.

Le moteur pour écrire un modèle s'appelle T4.

On y décrit les éléments suivants :

- recette : nom de la tâche que l'on souhaite automatiser, composée d'un ensemble d'actions séquencés
- wizard : l'assistant visuel qui permet à l'utilisateur de renseigner et paramétrer les actions

Lors du déploiement de votre package « GAT », vous devez installer les Guidance Automation Extensions (GAX) pour permettre à Visual Studio d'utiliser vos fabriques GAT.

GAT/GAX constituent des briques d'extensibilité pour Visual Studio et doit être utilisé dans la phase de développement de vos projets.

Pour faire le lien entre votre projet et le guidance package que vous avez installé sur votre poste, vous pouvez utiliser le Guidance Package Manager, le fichier gpstate ou créer votre projet sur la base du template proposé dans Visual Studio.

4 DSL, le designer de designer

Les Domain-Specific Language Tools vous permettent de créer vos propres « designer » afin de concevoir dans Visual Studio vos modèles métier.

L'objectif ici est d'exprimer votre métier dans Visual Studio dans un langage qui sera compris par vos clients et vous-même, dans l'optique de générer le squelette de vos applications.

En effet, après avoir élaboré et déployer votre DSL dans Visual Studio, vous pourrez modéliser votre application sur la base des éléments que vous avez préparé. Le modèle est sérialisé en XML dans un fichier avec l'extension dsl lors de son élaboration. Vous pourrez choisir une extension « maison » qui sera utilisé sur les postes où vous déploierez votre DSL.

On peut y ajouter des modèles GAT de génération de code qui peuvent être appelés depuis votre modèle DSL.

L'investissement dans la construction d'un DSL est profitable s' il est intégré dans une démarche globale d'industrialisation des développements. Si le métier est connu et spécialisé, il est intéressant d'investir dans un langage « maison » vous permettant de modéliser vos applications et d'y associer de la génération de code. Au-delà du contexte métier, vous pouvez utiliser les DSL également pour des modélisations orientés technique (tels les diagrammes Architectes proposés dans Visual Studio Team Suite).

Les DSL Tools sont intégrés au SDK de Visual Studio 2005.

5 VSIP / Add-In / Macro

La macro est le moyen le plus simple pour un développeur d'automatiser des tâches répétitives. Il peut enregistrer des séquences de son travail pour les rappeler ensuite en appuyant sur la combinaison de touches ALT + F11. Les limites sont que le seul langage supporté pour écrire une macro est le VB.NET et que le déploiement d'une bibliothèque de macros est limité au partage du projet qui contient le code source.

Ressource :

[http://msdn2.microsoft.com/en-us/library/b4c73967\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/b4c73967(VS.80).aspx)

Un Add-in Visual Studio permet à la manière d'une macro d'automatiser certaines tâches et possède l'avantage de pouvoir être compilé et ainsi moins permissible donc plus sécurisé.

Ressource :

[http://msdn2.microsoft.com/en-us/library/5abkeks7\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/5abkeks7(VS.80).aspx)

<http://www.visualstudiohacks.com/type/addins>

Visual Studio Integration Package (VSPackages ou VSIP Interfaces) vous permet d'atteindre directement le modèle objet de Visual Studio. Bon à savoir, si vous souhaitez déployer votre package, vous devez demander une PLK (Package Load Key) à partir du site Microsoft Visual Studio Industry Partner Program (<http://www.vsipmembers.com/>).

Ressource :

[http://msdn2.microsoft.com/en-gb/library/za2b25t3\(VS.80\).aspx](http://msdn2.microsoft.com/en-gb/library/za2b25t3(VS.80).aspx)

6 Retours d'expérience

Les architectures devenant plus complexes, il est impératif d'expliquer et de former à l'utilisation pertinente de ces outils afin de ne pas polluer le développeur avec l'apprentissage d'un nième framework.

De la même façon que le monde ne s'est pas fait en un jour, il est difficile d'élaborer sa fabrique logicielle en une seule étape. Ainsi, en observant l'utilisation de votre package (vous pourrez partir d'une fabrique existante), vous pourrez en déduire les nouveaux besoins en automatisation de code et enrichir votre package avec de nouvelles « recettes » (voir GAT). Le redéploiement n'impactera que le moteur de génération sous-jacent à votre solution et à vos projets.

Un simple projet de setup suffit à générer un MSI que vous déploierez sur les postes de développement et qui enregistrera votre package à la liste des templates de Visual Studio.

Développer une industrialisation logicielle nécessite une connaissance des outils à disposition ainsi qu'une analyse du contexte. L'architecte logiciel est la personne la plus à même à accomplir cette tâche en collaboration avec les développeurs.

7 Conclusion

Une multitude d'outils sont mis à notre disposition pour industrialiser nos développements. Pour s'y retrouver, il est nécessaire d'accompagner les développeurs sur le terrain et tenir compte des retours d'expérience en termes d'adoption et d'efficacité.

Si vous avez déjà entamé une démarche de capitalisation, vous vous trouvez confronté à la problématique de mise à disposition des composants et connaissances auprès de vos équipes techniques. Les outils Microsoft pour industrialiser les développements que nous avons vus dans ce document facilitent cette communication du savoir emmagasiné auprès des développeurs.

La démarche d'industrialisation des développements doit se faire progressivement avec des gains visibles. Une génération de code basée sur une modélisation compréhensible par le métier et la technique est un bon exemple. Gagner en productivité ne doit pas être la seule raison pour industrialiser les développements. Pour garantir la qualité, la fiabilité et la cohérence technologique de vos projets, miser sur l'automatisation des tâches de modélisation, de développement, de tests et de déploiement.

8 Table de références

Web Service Software Factory :

- <http://msdn.microsoft.com/msdnmag/issues/06/12/ServiceStation/Default.aspx?loc=fr>
- <http://msdn2.microsoft.com/en-us/library/aa480534.aspx>
- <http://www.codeplex.com/servicefactory>

Smart Client Software Factory :

- <http://msdn2.microsoft.com/en-us/library/aa480482.aspx>
- <http://www.codeplex.com/smartclient>

Web Client Software Factory :

- <http://msdn2.microsoft.com/fr-fr/library/bb264518.aspx>
- <http://msdn.microsoft.com/msdnmag/issues/07/08/ExtremeASPNET/default.aspx?loc=fr>
- <http://www.codeplex.com/websf>

Mobile Client Software Factory :

- <http://msdn2.microsoft.com/en-us/architecture/aa480471.aspx>
- <http://www.windowsfordevices.com/articles/AT8794680501.html>

DSL (Domain Specific Language) :

- <http://www.dslfactory.org>
- <http://msdn2.microsoft.com/en-us/vstudio/aa718368.aspx>
- <http://msdn2.microsoft.com/fr-fr/architecture/bb286889.aspx>
- <http://msdn2.microsoft.com/en-us/arcjournal/bb245773.aspx>

A propos d'Adiba

Adiba est une société française basée à Paris qui propose du conseil et du service en informatique spécialisé sur la plateforme Microsoft.

La mission d'Adiba est d'accompagner ses clients dans leur croissance au travers de la mise en œuvre d'outils d'industrialisation logicielle.

Adiba remercie les personnes qui ont participé directement ou indirectement à la rédaction de ce livre blanc, en particulier Stève Sfartz, architecte en système d'informations chez Microsoft qui a effectué la relecture de ce livre blanc.

<http://www.adiba.fr>

ADIBA

FACILITATEUR D'ARCHITECTURE