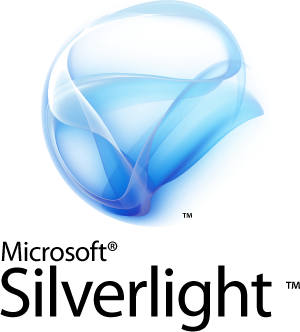
HappyNet

-

Application de gestion en Silverlight

[ Conseils & bonnes pratiques ]



****

|  |  |  |
| --- | --- | --- |
| **Liste de Distribution** | | |
|  | **Nom** | **Entité** |
| logo_so@t_entete CMJN | Codeplex  Consultants So@t  Direction Technique | DT |

|  |  |  |
| --- | --- | --- |
| **Validation** | | |
|  | **Nom** | **Date** |
| Auteur(s) | CATHALA Cyril | 11/02/2010 |
| Revue(s) |  |  |
|  |  |
|  |  |
| Validation (s) |  |  |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Suivi des versions** | | | |
| **Version** | **Date** | **Porteur** | **Description** |
| 1.0 | 11/02/2010 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**SOMMAIRE**

[1 Introduction 4](#_Toc261961358)

[1.1 Objet du document 4](#_Toc261961359)

[1.2 Pré-requis 4](#_Toc261961360)

[2 Présentation 5](#_Toc261961361)

[2.1 Outils de développement et composants logiciels 5](#_Toc261961362)

[2.2 Installation 5](#_Toc261961363)

[3 Architecture 7](#_Toc261961364)

[3.1 Architecture du code 7](#_Toc261961365)

[3.2 Architecture applicative 9](#_Toc261961366)

[4 MVVM : Rappel 10](#_Toc261961367)

[5 Points techniques 12](#_Toc261961368)

[5.1 PRISM & Navigation Framework 14](#_Toc261961369)

[5.2 Web services 18](#_Toc261961370)

[5.3 Localisation 24](#_Toc261961371)

[5.4 Authentification 25](#_Toc261961372)

[5.5 Logging 27](#_Toc261961373)

[5.6 Web Analytics 30](#_Toc261961374)

[5.7 Référencement 31](#_Toc261961375)

[5.8 Fuites mémoire 33](#_Toc261961376)

[5.9 Autres scénarios 34](#_Toc261961377)

[6 Conseils & Bonnes pratiques 39](#_Toc261961378)

[7 Composants externes 40](#_Toc261961379)

# Introduction

## Objet du document

La communauté Silverlight se porte maintenant très bien et se veut dynamique et regorgeant d’articles en tout genre. On y trouve des astuces, des contrôles personnalisés, des fonctionnalités passées au peigne fin, etc.

Cependant, lorsqu’il s’agit de développer une application concrète, on manque de piste sur de nombreux aspects du web pourtant classiques.

Comment construire une application Silverlight en environnement .NET ?

Comment harmoniser mon application riche avec d’autres concepts tels que le référencement, Google Analytics ou autres ?

C’est autant de questions auxquelles nous allons tenter de répondre à travers une application de « référence ». Le but est de proposer un ensemble de solutions répondant à des besoins fréquents lors de la construction d’applications en Silverlight.

Nous n’avons pas pour prétention de répondre à tous les problèmes, ou de la meilleure manière qui soit. C’est avant tout un retour d’expérience (fort de 2 ans d’expertise) donnant des pistes pour appréhender ce merveilleux monde de lumière d’argent …!

## Pré-requis

Cette documentation s’adresse aux développeurs .NET ayant des compétences avancées sur le développement d’application Silverlight et WCF.

# Présentation

L’application **HappyNet** se base sur la base de données d’exemple Microsoft AdventureWorks. Cette base de données stocke les informations de la société fictive **Adventure Works Cycles**.

Cette application comporte plusieurs parties distinctes, à savoir : la gestion des ressources humaines (schéma HumainResources), la gestion de la Production (schéma Production), la gestion des personnes (schéma Person) et la gestion des clients et des ventes (schéma Sales).

Il s’agit d’une application de gestion de type e-commerce on ne peut plus classique, dont l’objectif est de vendre ses produits à des utilisateurs authentifiés.

## Outils de développement et composants logiciels

* **Visual Studio 2008 SP1 \***

<http://www.microsoft.com/downloads/details.aspx?FamilyID=fbee1648-7106-44a7-9649-6d9f6d58056e&displaylang=fr>

* **Silverlight 3 Tools \***

<http://www.microsoft.com/Downloads/details.aspx?familyid=9442B0F2-7465-417A-88F3-5E7B5409E9DD&displaylang=en>

* **Composite Application Guidance Feb 2009** (PRISM v2.0)

<http://compositewpf.codeplex.com/>

* **Silverlight Toolkit November 09**

<http://silverlight.codeplex.com/releases/view/36060>

* **Entity Framework**
* **AdventureWorks 2008 \***

<http://www.codeplex.com/MSFTDBProdSamples>

* **Enterprise Library 4.1**

<http://www.microsoft.com/downloads/details.aspx?familyid=1643758B-2986-47F7-B529-3E41584B6CE5&displaylang=en>

* **Microsoft Practices Project Linker**

<http://www.microsoft.com/downloads/details.aspx?FamilyID=fa07e1ce-ca3f-4b9b-a21b-e3fa10d013dd&displaylang=en>

*Documentation :* <http://msdn.microsoft.com/en-us/library/dd458870.aspx>

\* : seuls les composants annotés sont à installer pour compiler et faire fonctionner l’application. Le reste des dll est embarqué avec l’application dans le répertoire /Assemblies.

## Installation

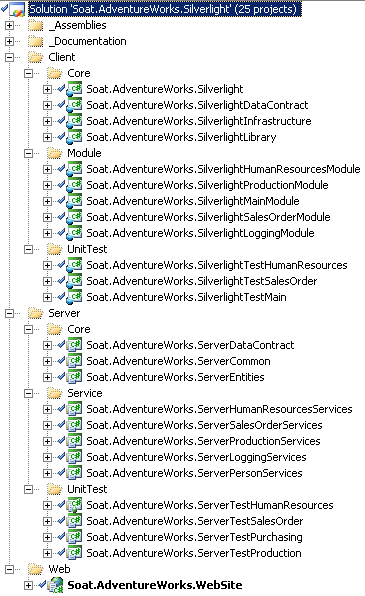
* Penser à créer un compte SQL Server pour accéder à la base et modifier le Web.config
* Exécuter les \*.sqldu répertoire « \_Documentation » afin de :
  + Ajouter un nouvel utilisateur (**login/password = demo/demo**)
  + Ajouter une procédure stockée pour récupérer la liste des employés

Et c’est tout !

# Architecture

L’application se compose de deux parties importantes, à savoir la partie cliente Silverlight et la partie Serveur, services métiers qui sont hébergés par une application Web ASP.NET.

## Architecture du code



Description de la structure du code :

* **Client** : Client Silverlight
  + **Core** : Projets communs à tout le client
    - **Soat.AdventureWorks.Silverlight** :

Projet « *Shell* », point d’entrée de l’application. Il contient l’App.xaml, le Bootstrapper propre à Prism (initialisation, configuration) et le RootVisual (appelé Shell dans le vocabulaire Prism).

* + - **Soat.AdventureWorks.SilverlightDataContract** :

Partage de données entre le client et le serveur (cf. ).

* + - **Soat.AdventureWorks.SilverlightInfrastructure :**

Projet partageant divers composants avec les modules (événements, strings, localisation, etc.).

* + - **Soat.AdventureWorks.SilverlightLibrary :**

Trousse à outils, contient des contrôles personnalisés, des helpers, des extensions …

* + **Modules :** Module indépendants, fondamental de Prism, qui assure un avantage sur de gros projets
    - **Soat.AdventureWorks.SilverlightHumanResourcesModule :**

Module de gestion des ressources humaines

* + - **Soat.AdventureWorks.SilverlightLoggingModule :**

Module de gestion des logs serveur

* + - **Soat.AdventureWorks.SilverlightMainModule :**

Module de gestion général, concerne tant la localisation que l’authentification ou autre …

* + - **Soat.AdventureWorks.SilverlightProductionModule :**

Module de gestion de la boutique et de ses produits

* + - **Soat.AdventureWorks.SilverlightSalesOrderModule :**

Module de gestion de la vente de produits (panier d’achat, paiement, etc.)

* + **UnitTest :** Tests unitaires client
* **Server** : Implémentation côté serveur
  + **Core :** Projets communs à tout le serveur
    - **Soat.AdventureWorks.ServerCommon :**

Composants communs au serveur (extensions, comportements …)

* + - **Soat.AdventureWorks.ServerDataContract :**

Partage de données entre le client et le serveur (cf. ).

* + - **Soat.AdventureWorks.ServerEntities :**

Schéma des entités généré avec l’EntityFramework

* + **Service :**
    - **Soat.AdventureWorks.ServerHumanResourcesServices :**

Définition du web service de gestion des ressources humaines

* + - **Soat.AdventureWorks.ServerLoggingServices :**

Définition du web service de gestion des logs serveur

* + - **Soat.AdventureWorks.ServerPersonServices :**

Définition du web service de gestion de personnes

* + - **Soat.AdventureWorks.ServerProductionServices :**

Définition du web service de gestion de la boutique

* + - **Soat.AdventureWorks.ServerSalesOrderServices :**

Définition du web service de gestion des ventes

* + **UnitTest :** Tests unitaires serveur
* **Web :** site web hébergeant l’application Silverlight

## Architecture applicative



# MVVM : Rappel

Le Model-View-ViewModel (MVVM) se résume à du Model-View-Controller (MVC) auquel on a ajouté le DataBinding, qui est l’élément clef de ce pattern.

Le principe est toujours le même : obtenir le plus faible couplage possible entre l’interface graphique et la logique métier de l’application.

Si on résume le rôle de chacun :

|  |  |  |
| --- | --- | --- |
| **View** | **ViewModel** | **Model** |
| **Interface utilisateur** | **Logique applicative** | **Persistence de données** |
| * + XAML   + Code-behind optionnel | * + Etat de la Vue (propriétés)   + Opérations (méthodes)   + Notifications (événements) | * Couche d’accès aux données |

**Avant**(c’est mal !) **:**

**Après** (c’est bien !)**:**

Plus besoin de tester la vue et ses innombrables interactions utilisateur, ce qui nous intéresse ici est la logique métier, contenue dans le ViewModel.

Grâce à ce pattern, on obtient donc un code plus facile à écrire, à lire, et donc à maintenir et tester.

# Points techniques

|  |  |  |
| --- | --- | --- |
| Problématique | Fonctionnalité HappyNet | Exemple de démonstration : |
| Comment harmoniser le système de région de PRISM et le Navigation Framework ? | Navigation | * Soat.Silverlight/Shell.xaml * Soat.Silverlight.Module/ViewPages/\*Page.xaml |
| Quand mon web service envoie une FaultException, j’ai une erreur 404 côté Silverlight ! | Behavior WCF utilisé côté serveur | * Soat.ServerCommon/SilverlightFaultBehavior/\* * Soat.WebSite/Web.config |
| Client Http Stack | * Soat.Silverlight/App.xaml.cs |
| Mes proxies web services Visual Studio ne partagent pas leurs définitions de classes ! | Partage du projet DataContract | * Soat.SilverlightDataContract * Soat.ServerDataContract |
| Découpage logique en modules | * Soat.SilverlightModule/Services/\* |
| J’ai créé plusieurs proxies VS dans différents modules et ils ne trouvent pas leur configuration ! | Centralisation de la configuration et encapsulation des web services | * Soat.Silverlight/ServiceReferences.ClientConfig * Soat.SilverlightModule/Services/\* |
| Comment changer de localisation dynamiquement, sans rafraîchir la page ? | Localisation dynamique | * Soat.SilverlightInfrastructure/Localization/\* * Soat.SilverlightInfrastructure/Localization/   LocalizedStrings.cs   * Soat.SilverlightMainModule/Views/   LanguageSwitch/\* |
| Comment gérer l’authentification utilisateur en Silverlight ? | Authentification requise pour accéder à la liste des employés | * Soat.SilverlightInfrastructure/Helpers/   OperationContextHelper.cs   * Soat.SilverlightHumanResourcesModule/Services/HumanResourcesService.cs * Soat.ServerPersonServices/Authentication/\* * Soat.HumanResourcesServices/ServiceLayer/   HumanResourcesService.cs |
| Silverlight s’exécute côté client, comment récupérer des logs lorsque mon application plante ? | Système de logging serveur | * Soat.ServerLoggingServices * Soat.SilverlightLoggingModule |
| Comment faire pour avoir des statistiques sur les actions de mes utilisateurs ? | Tracking via Google Analytics | * Soat.SilverlightLibrary/Behaviors.Actions/   TrackEventAction.cs   * Soat.SilverlightProductionModule/Views/   ProductsList/ProductItem/ProductItemView.xaml |
| On me dit souvent que les technologies RIA sont mal référencées … | Référencement via pages alternatives | * Soat.WebSite * Soat.WebSite/App\_Data/PageMappings.xml * Soat.WebSite/AppCode/XmlPageMappings.cs |
| Aïe! J’ai une fuite mémoire, impossible de trouver d’où ça vient ! | WinDbg |  |
| Comment partager un composant d’un module avec toute mon application ? | Bouton du menu de navigation | * Soat.SilverlightInfrastructure/Interfaces/   IMenuItemView.cs   * Soat.SilverlightInfrastructure/Interfaces/   IMenuItemViewModel.cs   * Soat.SilverlightMainModule/Views/MenuItem/   MenuItemView.xaml   * Soat.SilverlightMainModule/Views/MenuItem/   MenuItemViewModel.cs |
| Comment binder mon bouton à une commande de mon ViewModel quand je suis à l’intérieur d’un DataTemplate ? |  |  |
| Quand je bind quelque chose à mon header, ça plante ! | En-têtes localisées du panier d’achat | * Soat.SilverlightSalesOrderModule/Views/   ShoppingCart/ShoppingCartView.xaml |
| Dans mon CellTemplate, je n’arrive pas à binder mon bouton à la commande de mon ViewModel ! | Bouton de suppression d’un produit du panier d’achats | * Soat.SilverlightSalesOrderModule/Views/   ShoppingCart/ShoppingCartView.xaml |
| Quand j’essaie de changer l’url, ça me recharge systématiquement la page, alors que je ne veux pas ! | Modification de l’url de navigation sans naviguer | * Soat.SilverlightInfrastructure/Globals.cs * Soat.Silverlight/Shell.xaml.cs |
| Quand un utilisateur clique rapidement sur tous mes liens, ça charge les données n’importe comment ! | Affichage des produits d’une catégorie | * Soat.SilverlightProductionModule/Views/   ProductsList/ProductsListViewModel.cs |
| Je ne peux sélectionner aucun texte dans mon application ! Comment je fais mon copié/collé ?! | Description du détail d’un produit sélectionnable | * Soat.SilverlightProductionModule/Views/   ProductDetail/ProductDetailView.xaml |

## PRISM & Navigation Framework

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Comment harmoniser le système de région de PRISM et le Navigation Framework ? | **Solution :**   * Utiliser des Page pour naviguer et le « View Discovery » de Prism pour associer les vues aux régions |

**PRISM ?**

Composite Application Library (CAL) ou Prism v2.0 du « Microsoft patterns & practices Composite Application Guidance for WPF and Silverlight », qu’est-ce donc ?!

Le maître mot autour de ce framework est **découplage**.

Il fournit des outils et concepts tels que :

* Architecture du code
* Modularité
* Injection de dépendances
* Communication par médiateur
* Composition de vues
* Aide au développement MVVM

Si vous ne connaissez pas, il est encore temps de le découvrir (vite !). Le MSDN fourmille déjà de schémas et documentation en tout genre :

<http://msdn.microsoft.com/en-us/library/dd458809.aspx>

Pour faire un tour rapide de la question, il existe des screencasts très bien faits (mais en anglais) :

* [Introduction à Prism](http://development-guides.silverbaylabs.org/Video/Silverlight-Prism)
* [Modularité sous Prism](http://development-guides.silverbaylabs.org/Video/Modularity-in-Prism)
* [Gestion des régions en Prism](http://development-guides.silverbaylabs.org/Video/Prism-Regions)
* [Commanding sous Prism](http://development-guides.silverbaylabs.org/Video/Prism-Commands)

**Composition des vues ?**

Pour notre problème, ce qui nous intéresse est la fonctionnalité de « UI Composition ». Le principe reprend grosso modo les MasterPage de l’ASP.NET.

On dessine une vue, on y insère des conteneurs (ContentControl, ItemsControl, etc.) auxquels on assigne un nom de région. Dès lors, ce sont des conteneurs de région dans lesquels on peut injecter une vue de manière dynamique.

Pour donner un exemple de code, voilà comment définir un conteneur de région :

<!-- Conteneur XAML de région -->

<ContentControl regions:RegionManager.RegionName="ProductsRegion"

x:Name="ProductsContainer"/>

Et voici comment y injecter une vue, dès que le conteneur de région est chargé :

// Enregistrement de la vue dans la région

this.regionManager.RegisterViewRegion("ProductsRegion", typeof(IProductsListView));

Comme rien ne vaut un bon schéma, tentons de résumer le principe :

Ici j’ai donc une Page contenant 2 régions : NavigationRegion et ProductRegion.

NavigationView est associée à la NavigationRegion.

ProductsView est associé à la ProductRegion.

Dès que la Page est chargée (et donc ses régions), les vues sont injectées dans les régions associées.

La méthode d’enregistrement des vues décrite ici est appelée le « View Discovery ».

**Limite du Navigation Framework ?**

La limite du Navigation Framework provient du fait qu’il n’est pas possible (en Silverlight 3) de mapper une URI vers autre chose qu’un fichier .xaml de type Page. Idéalement, on aurait souhaité rediriger directement vers une View instanciée par notre conteneur IoC.

La solution consiste donc à utiliser une Page intermédiaire, contenant toutes les régions :

A une adresse du UriMapper (ancre #) correspond une Page (.xaml), qui contient des régions, chacune liée à une Vue.

|  |  |
| --- | --- |
| http://www.clker.com/cliparts/c/1/a/7/11954229571856793418warning_naught101_01.svg.med.png | **Problème de compatibilité :**  Si on navigue vers une page contenant des régions, qu’on la quitte, puis qu’on revient sur la même page, une exception est levée car les régions de la page ont déjà été créées.  **Solution :**  Pour résoudre ce soucis, une solution est de nettoyer les régions à chaque navigation.  Cf. la méthode ClearRegionManager() dans Soat.Silverlight/Shell.xaml.cs. |

|  |  |
| --- | --- |
| http://www.clipart-fr.com/data/icones/Applications/icones_01104.png | **Idée :**  A noter par rapport au warning précédent que la méthode ClearRegionManager() permet également de nettoyer les ViewModels de l’ancienne page lorsqu’on navigue. Pour ce faire, on implémente le IDisposable ou une classe personnelle de type ICleanable. |

**N.B. :** A noter que la DLL *Microsoft.Practices.Composite.Presentation.dll* embarquée contient un correctif qui, couplé au ClearRegionManager, évite certaines exceptions. Pour information, le correctif est le suivant :

// Dans DelayedRegionCreationBehavior, dans la méthode TryCreateRegion :

private void TryCreateRegion()

{

[..]

if (targetElement.CheckAccess())

{

this.Detach();

if (!this.regionCreated)

{

// Booléen à placer juste après le if,

// pour éviter 2 appels consécutifs à CreateRegion

this.regionCreated = true;

string regionName = this.RegionManagerAccessor.GetRegionName(targetElement);

CreateRegion(targetElement, regionName);

}

}

}

Finalement, bien que le RegionManager simplifie bien des choses, s’il n’est pas prudemment utilisé, il apporte plus de problèmes qu’autre chose …

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Navigation | **Exemple de démonstration :**   * Soat.Silverlight/Shell.xaml * Soat.Silverlight.Module/ViewPages/\*Page.xaml |

## Web services

### Communication

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Quand mon web service envoie une FaultException, j’ai une erreur 404 côté Silverlight ! | **Solution :**   * WCFBehavior * Http Client Stack |

**Quelques explications …**

Par défaut, lorsqu’un web service renvoie une FaultException, le code de statut HTTP envoyé en en-tête de la réponse est égal à **500**.

Chaque navigateur limite le networking stack assigné à leurs plugins. Ainsi, les plugins navigateur tels que Silverlight ne peuvent recevoir que des statuts **200** (tout est OK), ou **404** (le fameux Not Found).

Donc lorsqu’un web service envoie une FaultException, le navigateur récupère bien un code **500** (que l’on peut intercepter grâce à un sniffer type Fiddler ou Firebug), mais le plugin lui, ne reçoit qu’un **404** dépourvu de tout détail car filtré par le navigateur.

**Que faire ?!**

* En Silverlight 2.0 :

On est limité à la pile réseau du navigateur, il faut donc trouver un moyen d’encapsuler les FaultExceptions sous un code **200**. Car heureusement pour nous, le client Silverlight sait gérer le contenu d’une FaultException, il suffit juste de rendre ce contenu accessible.

Si les web services sont en WCF, nous sommes sauvés ! Il suffit dès lors d’y appliquer un WCFBehavior pour rediriger toutes les FaultException en HTTP Status Code 200.

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Behavior WCF utilisé côté serveur | **Exemple de démonstration :**   * Soat.ServerCommon/SilverlightFaultBehavior/\* * Soat.WebSite/Web.config |

* En Silverlight 3.0 :

Bonne nouvelle, on n’est plus limité au Browser Stack ! Nouveauté Silverlight 3.0 : le Client Stack.

Celui-ci passe non plus par la pile réseau du navigateur, mais par celle de l’OS. Il supporte tous les statuts http, plus besoin de retoucher aux web services !

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Client Http Stack | **Exemple de démonstration :**   * Soat.Silverlight/App.xaml.cs |

On a plusieurs méthodes pour l’utiliser.

* + Soit on choisit d’enregistrer un préfixe d’url, auquel cas tous les appels vers ce préfixe se feront via le stack sélectionné :

// Activation du Client networking stack, supportant les FaultExceptions

HttpWebRequest.RegisterPrefix("http://", WebRequestCreator.ClientHttp);

HttpWebRequest.RegisterPrefix("https://", WebRequestCreator.BrowserHttp);

* + Soit on configure le choix du stack à chaque appel :

// Configuration par appel

HttpWebRequest request = (HttpWebRequest)WebRequestCreator.ClientHttp.Create(

new Uri("http://www.monsite.com", UriKind.Absolute));

Dans tous les cas, on notera les quelques différences entre le browser stack et le client stack. Tous deux ont leurs avantages et inconvénients …

|  |  |  |
| --- | --- | --- |
| **Silverlight Networking Stack** | | |
| **Fonctionnalité** | **Browser Stack** | **Client Stack** |
| Authentification | Géré par le browser | Non supporté |
| Cache | Géré par le browser | Non supporté |
| Cookies | Géré par le browser | Gérable, mais non partagé avec browser |
| Méthodes HTTP | GET & POST | Toutes (selon autorisation) |
| Proxy | Configuration proxy browser | Configuration proxy OS |
| En-têtes requête | Requêtes GET | Toutes les requêtes |
| En-tête réponse | Non supporté | Supporté sur HttpWebResponse (pas WebClient) |
| **Code statut HTTP** | **200 et 404 uniquement** | **Tous** |

Pour plus d’informations :

* [Client / Browser networking stacks](http://www.wintellect.com/CS/blogs/jprosise/archive/2009/10/14/silverlight-3-s-new-client-networking-stack.aspx)
* [WCF Faulting Behavior](http://msdn.microsoft.com/en-us/library/dd470096(VS.96).aspx)

### Accès aux données

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Mes proxies web services Visual Studio ne partagent pas leurs définitions de classes ! | **Solution :**   * Partager les classes serveur avec le client |

**Visual Studio et la génération de proxies …**

Lorsqu’on travaille avec des web services et un outil comme Visual Studio, on a tendance à générer un proxy web services (*Service Reference*) côté client pour accéder aux web services exposés par un serveur quelconque.

Certains vous diront que le proxy génère du code sale, qu’il vaut mieux les implémenter à la main comme grand-père. C’est effectivement plus propre, mais qui a envie de gérer les accès concurrents, écrire toutes les classes DTO, implémenter le pattern asynchrone, etc. …?

Les développeurs tâchent avant tout de ne pas réinventer la roue en étant productifs et efficaces (et quelques peu fainéants, certes).

Bref, nous utiliserons donc le proxy généré par Visual Studio …

|  |  |
| --- | --- |
| http://www.clker.com/cliparts/c/1/a/7/11954229571856793418warning_naught101_01.svg.med.png | **Problème :**  Avez-vous déjà essayé de générer plusieurs proxies partageant certaines classes ?  Si oui, vous connaissez certainement le problème traité ici : les proxies ne fusionnent par leurs définitions de classe. Si le proxy gérant les stocks et le proxy gérant les ventes utilisent la même classe Produit, on obtiendra bien 2 classes distinctes, chacune définie dans 2 namespaces différents. |

|  |  |
| --- | --- |
| http://www.clipart-fr.com/data/icones/Applications/icones_01104.png | **Solution :**  La seule solution proposée par Visual Studio est de référencer une dll qui contient la définition des classes du web services, avec le même namespace. |

Idéalement, on aurait donc besoin d’un projet à part, contenant toutes nos classes. Ainsi, on pourrait le référencer et éviter les doublons générés par les proxies Visual Studio.

Poussons encore plus loin : et si on partageait directement les classes définies côté serveur (web service) avec notre client (Silverlight) ?

Nos classes vont dépendre de notre manière d’accéder aux données. Voyons différents cas de figure …

1. **Voie classique**

Dans ce cas, on accède à la base de données via ADO.NET, procédures stockées, en s’aidant par exemple de l’Enterprise Library.

|  |  |
| --- | --- |
| http://www.clipart-fr.com/data/icones/Applications/icones_01104.png | Dans ce cas de figure, la solution est plutôt simple : créer un projet contenant toutes nos classes, qui sera ensuite partagé avec le client. |

|  |  |
| --- | --- |
| http://www.clker.com/cliparts/c/1/a/7/11954229571856793418warning_naught101_01.svg.med.png | Un projet Silverlight ne peut référencer que des projets basés sur la même CLR, allégée par rapport au framework .NET. |

L’astuce consiste donc en la création d’une **Silverlight Class Library** qui sera « liée » aux classes de notre projet serveur.

Pour créer ce lien, on utilisera le [ProjectLinker](http://msdn.microsoft.com/en-us/library/dd458870.aspx) livré avec la package Prism.

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Partage du projet DataContract | **Exemple de démonstration :**   * Soat.SilverlightDataContract * Soat.ServerDataContract |

1. **Voie moderne**

Le framework .NET a évolué et nous propose en 3.5 d’utiliser l’EntityFramework.

Pourquoi ne pas utiliser la même méthode que précédemment, à savoir créer un projet et le lier au client Silverlight ?

|  |  |
| --- | --- |
| http://www.clker.com/cliparts/c/1/a/7/11954229571856793418warning_naught101_01.svg.med.png | **Problème :**  Les entités générées héritent de EntityObject, qui fait appel à des DLL non supportées en Silverlight. On a donc un problème de compatibilité.  N.B. : Ce problème n’a plus lieu d’être avec EntityFramework 4. |

Doit-on créer des DTO qui viendraient en surcouche des entités ?

Ce serait pratique pour abstraire les entités et moins coller à la base de données. Mais ça revient à faire du recopiage de classe, c’est long et laborieux …

Doit-on créer des DTO côté client directement ?

Ces objets viendraient abstraire les classes générées par le proxy, ce qui ne serait pas un mal en soi, on aurait moins de dépendance. Seulement encore une fois, c’est un travail long et pénible …

|  |  |
| --- | --- |
| http://www.clipart-fr.com/data/icones/Applications/icones_01104.png | Il n’y a pas de solution miracle ou optimale. L’approche qui a été utilisée dans HappyNet est de procéder à un découpage logique entre modules et services. |

Voici un exemple d’implémentation basé sur des web services de provenances multiples. Le principe étant que lorsqu’on souhaite communiquer des objets entre modules, on passera par leur identifiant et non l’objet concret lui-même.

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Découpage logique en modules | **Exemple de démonstration :**   * Soat.SilverlightModule/Services/\* |

1. **.NET RIA Services**

A noter que RIA Services permet de générer automatiquement les DTO en surcouches des entités, et partage les classes directement avec le client. Pourquoi ne pas l’utiliser ici ?

|  |  |
| --- | --- |
| **Plus** | **Moins** |
| Génère les DTOs et les partage avec le client | Version beta |
| Facilitation de l’aspect asynchrone | Demande des compétences supplémentaires |
| Gain en productivité dans la plupart des cas | Pas d’interopérabilité hors .NET |
|  | « Magique », difficile à personnaliser |

On retiendra au final que cette technologie dépend des besoins et de l’existant. Au demeurant, cela ressemble quand même énormément aux DataSources introduites en ASP.NET 2.0, qui faisaient passer la programmation pour du simple drag&drop. Hum … « wait & see » comme on dit.

### Côté client

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + J’ai créé plusieurs proxies VS dans différents modules et ils ne trouvent pas leur configuration ! | **Solution :**   * Configuration centralisée dans le Shell Project |

Lorsqu’on génère un proxy avec Visual Studio dans un module donné, un fichier de configuration est généré : ServiceReferences.ClientConfig.

Aussi, quand on exécute notre application, le fichier de configuration est recherché non pas dans le projet du Module, mais dans le projet d’entrée de l’application, à savoir le Shell Project. L’idée est donc de centraliser la configuration dans le Shell Project.

Aussi, on notera qu’on n’est pas très flexible par rapport aux proxies créés. De manière à abstraire un minimum cette couche, on va l’encapsuler.

**http://www.clipart-fr.com/data/icones/Applications/icones_01104.png Le UserState, kézako ?**

Le proxy Visual Studio génère des méthodes asynchrones ayant 2 signatures : l’une correspondant à la signature du web service, l’autre avec un argument supplémentaire, le UserState.

Le UserState est un objet stocké côté client ! Ainsi, lorsqu’on l’assigne à l’appel du web service, on peut le récupérer au retour de celui-ci. Sachant qu’on travaille en mode asynchrone, c’est plutôt confortable !

Pour voir un exemple d’utilisation, consulter une classe Service d’un module d’HappyNet (par exemple ProductionModule/Services/ProductionService.cs).

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Centralisation de la configuration et encapsulation des web services | **Exemple de démonstration :**   * Soat.Silverlight/ServiceReferences.ClientConfig * Soat.SilverlightModule/Services/\* |

## Localisation

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Comment changer de localisation dynamiquement, sans rafraîchir la page ? | **Solution :**   * Binding sur objet implémentant *INotifyPropertyChanged* |

La méthode classique en .NET pour localiser son application : les fichiers de ressources (\*.resx). On peut bien sûr les utiliser en Silverlight :

<!-- Ressource de localization -->

<lang:StringLibrary x:Key="LocStrings" />

<TextBlock Text="{Binding Source={StaticResource LocStrings}, Path=WelcomeMessage}" />

|  |  |
| --- | --- |
| http://www.clker.com/cliparts/c/1/a/7/11954229571856793418warning_naught101_01.svg.med.png | **Problème :**  Lorsqu’on change la CultureUI de l’application, on constate que l’interface ne répercute pas le changement de langue, pour la simple et bonne raison qu’un fichier ressource n’implémente pas le INotifyPropertyChanged. |

|  |  |
| --- | --- |
| http://www.clipart-fr.com/data/icones/Applications/icones_01104.png | Pour mettre à profit le Binding, il faut passer par une classe implémentant INotifyPropertyChanged. |

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Localisation dynamique | **Exemple de démonstration :**   * Soat.SilverlightInfrastructure/Localization/\* * Soat.SilverlightInfrastructure/Localization/   LocalizedStrings.cs   * Soat. SilverlightMainModule/Views/   LanguageSwitch/\* |

## Authentification

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Comment gérer l’authentification utilisateur en Silverlight ? | **Solution :**   * Compatibilité ASP.NET * Messages Headers |

Plusieurs solutions disponibles pour l’authentification en Silverlight :

1. **Compatibilité ASP.NET**

WCF n’intègre pas l’authentification, la session et bien d’autres fonctionnalités propres aux applications web ASP.NET.

Heureusement, le framework .NET nous fournit un service WCF d’authentification : AuthenticationService. Celui-ci implémente un certain nombre de méthodes qui permettent l’authentification basée sur l’ASP.NET Forms Authentication.

On notera que cette méthode utilise les cookies pour stocker les informations utilisateur.

|  |  |
| --- | --- |
| http://www.clker.com/cliparts/c/1/a/7/11954229571856793418warning_naught101_01.svg.med.png | **Problème :**  Silverlight ne peut pas détecter l’expiration d’un cookie, stocké par le navigateur. |

On pourrait re-authentifier l’utilisateur à chaque requête WCF, ou s’amuser (c’est un bien grand mot) à personnaliser les cookies. Seulement voilà, cela devient plutôt compliqué …

Cette solution est donc à préconiser pour les connaisseurs, qui auront bien conscience de sa limite.

Pour plus d’informations à ce sujet :

* <http://blogs.dotnet-france.com/juliend/post/WCF-Authentification-sur-un-service-WCF-(compatible-Silverlight).aspx>
* <http://msdn.microsoft.com/en-us/library/bb386582.aspx>
* <http://www.silverlight.net/learn/videos/all/use-aspnet-authentication-silverlight-3/>

1. **En-têtes de messages**

Une autre façon de faire : injecter les identifiants de l’utilisateur dans chaque appel web service.

Nous sommes de petits veinards, les en-têtes de messages SOAP sont accessibles en Silverlight, et on peut y injecter ce qu’on veut !

Le principe est le suivant : à chaque appel web service, le client injecte dans les headers le couple login/password (ou un token généré au préalable), le serveur réceptionne, vérifie la validité des credentials, et exécute sa méthode si tout est ok.

De plus, côté serveur, il est possible d’encapsuler la lecture des en-têtes dans un attribut WCF. Pratique, on n’a plus qu’à tagger les méthodes qui nécessitent une authentification de l’utilisateur appelant, et c’est dans la poche.

Oui mais … on n’a plus de persistance de l’authentification (type cookie), et on doit vérifier le token (ou login/password) à chaque appel, ce qui est légèrement coûteux. Un moindre mal face à la facilité d’implémentation dirons-nous.

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Authentification requise pour accéder à la liste des employés | **Exemple de démonstration :**   * Soat.SilverlightInfrastructure/Helpers/   OperationContextHelper.cs   * Soat. SilverlightHumanResourcesModule/Services/   HumanResourcesService.cs   * Soat.ServerPersonServices/Authentication/\* * Soat.HumanResourcesServices/ServiceLayer/   HumanResourcesService.cs |

## Logging

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Silverlight s’exécute côté client, comment récupérer des logs lorsque mon application plante ? | **Solution :**   * Web services * CLog |

1. **Web service unidirectionnel**

Le plus simple est tout simplement d’implémenter un web service « one way », unidirectionnel, qui ne renvoie pas de donnée.

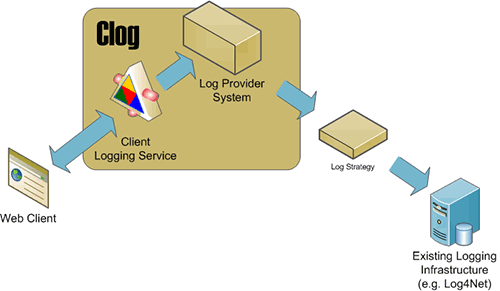
Ainsi, dès que le client Silverlight a un message à logger, il l’envoie, sans attendre de réponse.

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Système de logging serveur | **Exemple de démonstration :**   * Soat.ServerLoggingServices * Soat.SilverlightLoggingModule |

1. **CLog**

CLog est une solution disponible en open source qui vient s’interfacer avec un système de logging déjà existant. Il créé un web service WCF et s’intègre à la configuration déjà en place, plutôt pratique.

Plus d’informations sur : <http://clog.codeplex.com>.



1. **IsolatedStorage**

Scénario plus complexe : en mode déconnecté ou lorsque l’application plante sans pouvoir envoyer de données via web services.

Dans ce cas, on peut intégrer un modèle de reporting différé à la Windows.

L’application plante et/ou ne peut pas communiquer avec le serveur : on stocke nos logs dans l’IsolatedStorage.

1 : L’utilisateur relance l’application.

2 : On cherche des logs dans l’IsolatedStorage.

3 : Si on trouve des logs, on les récupère en pensant à les supprimer de l’IsolatedStorage, pour libérer l’espace.

4 : On envoie les logs au serveur via web services.

## Web Analytics

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Comment faire pour avoir des statistiques sur les actions de mes utilisateurs ? | **Solution :**   * Trigger / Action + HtmlBridge |

Dans une application web, il est toujours intéressant d’avoir des statistiques sur les actions utilisateurs. Comment rendre des outils comme Google Analytics disponible dans une application Silverlight ?

Silverlight peut interagir avec le Javascript via son « Html Bridge ». C’est là-dessus que nous nous baserons.

Il suffit d’intégrer Google Analytics à notre site web hébergeant le client Silverlight. Depuis Silverlight, on fera un appel à une méthode Javascript qui elle-même appelle Google Analytics.

**Triggers … Action !**

Blend nous livre une magnifique DLL souvent peu connue : System.Windows.Interactivity.dll.

Là-dedans réside des fonctionnalités fort utiles, j’ai nominé : les Triggers, Actions et Behaviors.

Un **Trigger**, c’est un événement déclencheur, auquel on lie une **Action**.

Ex. : Trigger se déclenchant au « MouseLeftButtonDown », Action associée : exécute une commande.

Lorsque le couple Trigger/Action n’est pas suffisant, on utilisera un **Behavior**. On pourra l’appliquer à n’importe quel composant, et il embarquera en quelque sorte un ensemble de Triggers et d’Actions.

Ex. : Ajouter un comportement de type drag&drop à un composant.

**TrackEventAction**

On le sentait venir … Grâce aux Triggers qui sont infinis, il suffit d’implémenter une Action qui appellera Google Analytics pour avoir une infinité de possibilité pour tracker les moindres faits et gestes des utilisateurs. Big Brother est à notre portée !

|  |  |
| --- | --- |
| http://www.clipart-fr.com/data/icones/Applications/icones_01104.png | A noter la mise à disposition par Microsoft d’un framework Analytics : [http://msaf.codeplex.com](http://msaf.codeplex.com/). |

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Tracking via Google Analytics | **Exemple de démonstration :**   * Soat. SilverlightLibrary/Behaviors.Actions/   TrackEventAction.cs   * Soat.SilverlightProductionModule/Views/ ProductsList/ProductItem/ProductItemView.xaml |

## Référencement

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + On me dit souvent que les technologies RIA sont mal référencées … | **Solution :**   * DeepLinking * Contenu alternatif |

Le problème de Silverlight, au même titre que Flash, est que le contenu n’est pas (ou partiellement) parsable par les moteurs de recherche.

L’idée est donc de présenter des données lisibles pour un robot, ce qui correspond à un utilisateur n’ayant pas installé le plugin Silverlight.

On va s’aider du Navigation Framework et de sa navigation par ancres (DeepLinking). Le soucis est que les ancres sont exécutées côté client, et ne sont donc jamais indexées par les moteurs de recherche.

La solution est alors d’utiliser des variables GET (ou de l’url rewriting, même principe).

Un robot de moteur de recherche arrive sur notre site. Première chose qu’il cherche : le robots.txt.

Ce dernier fichier va ensuite renseigner le robot sur la localisation du Sitemap. Dans notre cas, il s’agit de Sitemap.aspx qui contient un listing de tous les liens de notre application.

A chacune des pages ASPX correspond une page Silverlight. Aussi, on devra s’efforcer d’afficher le même contenu dans les 2 versions. L’une lisible par le moteur de recherche, l’autre par l’utilisateur souhaitant une expérience enrichie.

Maintenant, nous sommes un utilisateur avisé ayant déjà installé le plugin Silverlight.

Dès lors, on récupère l’url et on parse la QueryString (ici : « page=Products »).

On administre un fichier de mapping, qui nous donne la correspondance entre une url dite Html, et une url Silverlight (sous forme d’ancre, sur laquelle se base le NavigationFramework).

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Référencement via pages alternatives | **Exemple de démonstration :**   * Soat.WebSite * Soat.WebSite/App\_Data/PageMappings.xml * Soat.WebSite/AppCode/XmlPageMappings.cs |

## Fuites mémoire

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Aïe! J’ai une fuite mémoire, impossible de trouver d’où ça vient ! | **Solution :**   * WinDbg ! |

Les RIA relayent les mêmes problématiques que des clients lourds. On a une persistance de l’application côté client, et donc des fuites de mémoire potentielles. C’est souvent au développeur d’être suffisamment rigoureux pour pallier à ces soucis.

On fera donc attention aux références fortes empêchant le Garbage Collector de faire son travail (event handlers, références fortes, etc.).

Aussi, il existe quelques outils forts utiles afin de mieux tracker les fuites de mémoire :

* [ANTS Profiler](http://www.red-gate.com/products/dotnet_tools_overview.htm) (payant)

Très performant, mais ne fonctionne malheureusement qu’à partir de Silverlight 4.

* [WinDbg](http://www.microsoft.com/whdc/Devtools/Debugging/default.mspx) (gratuit !)

Très utile, mais l’interface n’est pas ce qu’il y a de plus explicite (vive la ligne de commande) …

Plutôt que refaire un tutoriel, je vous livre un article très bien rédigé qui fait un bon tour d’horizon des possibilités de WinDbg :

<http://blogs.msdn.com/delay/archive/2009/03/11/where-s-your-leak-at-using-windbg-sos-and-gcroot-to-diagnose-a-net-memory-leak.aspx>

En cas de doute quant à une fuite mémoire, je vous recommande très fortement cet outil !

**Quid des performances ...?**

Plutôt qu’un long discours, je vous donne directement les ressources intéressantes :

Screencast très complet sur le fonctionnement de Silverlight :

<http://microsoftpdc.com/Sessions/P09-10>

SilverlightSpy, outil d’inspection des applications Silverlight :

<http://silverlightspy.com/silverlightspy/download-silverlight-spy/>

Xperf pour visualiser les performances :

<http://blogs.msdn.com/seema/archive/2008/10/08/xperf-a-cpu-sampler-for-silverlight.aspx>

## Autres scénarios

### Interface globale (IMenuItem)

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Comment partager un composant d’un module avec toute mon application ? | **Solution :**   * Définir une interface accessible depuis l’Infrastructure |

Il est parfois plus logique d’implémenter une vue dans un module particulier. Cependant, on peut se rendre compte que cette vue pourrait être utilisée dans d’autres modules également.

Dilemme ! A-t-on mal découpé nos modules ? S’agit-il d’un cas à part ? La vue doit-elle être localisée dans le projet Infrastructure ?

La réponse dépendra des cas. Une « astuce » consiste en tout cas à localiser les interfaces dans le projet Infrastructure. Ainsi, les composants restent accessibles depuis tous les modules.

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Bouton du menu de navigation | **Exemple de démonstration :**   * Soat.SilverlightInfrastructure/Interfaces/IMenuItemView.cs * Soat.SilverlightInfrastructure/Interfaces/IMenuItemViewModel.cs * Soat.SilverlightMainModule/Views/MenuItem/MenuItemView.xaml * Soat.SilverlightMainModule/Views/MenuItem/MenuItemViewModel.cs |

### Binding sur ViewModel à l’intérieur d’un DataTemplate

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Comment binder mon bouton à une commande de mon ViewModel quand je suis à l’intérieur d’un DataTemplate ? | **Solution :**   * Utiliser l’ElementName |

Le Pattern MVVM apporte un meilleur découpage, une meilleure logique, mais peut parfois être contraignant …

Prenons un exemple :

<ItemsControl ItemsSource="{Binding DataList}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<StackPanel>

<TextBlock Text="{Binding Name}" />

<Button Content="Click me !"

cal:Click.Command="{Binding ClickCommand}" />

</StackPanel>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

Exemple basique où on itère sur une liste de données, et on affiche un libellé et un bouton. Le principe est d’assigner au bouton une commande définie dans le ViewModel.

Le problème est que dans le template, le DataContext du bouton est assigné à l’itération courante de DataList !

L’astuce pour retrouver le lien vers le ViewModel est d’utiliser l’attribut ElementName, pour cibler le DataContext d’un élément visuel plus haut dans l’arborescence, en dehors du template.

On peut par exemple utiliser le LayoutRoot :

<Button Content="Click me !"

cal:Click.Command="{Binding DataContext.ClickCommand, ElementName=LayoutRoot}" />

### Binding sur les Headers d’une DataGrid

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Quand je bind quelque chose à mon header, ça plante ! | **Solution :**   * Utiliser le ContentTemplate |

Lorsqu’on utilise une DataGrid dans une application localisée, il paraît logique de binder un libellé localisé à chaque en-tête de colonne. Malheureusement pour nous, la propriété Header des DataGridColumn ne supporte pas le DataBinding …

Pour pallier à ce souci, on passera par le ContentTemplate du style défini sur les headers.

Donc plutôt que d’avoir ça :

<data:DataGridTemplateColumn Header="{Binding Strings.SALES\_Quantity, Source={StaticResource LocalizedStrings}}">

[..]

</data:DataGridTemplateColumn>

On aura ceci :

<data:DataGridTemplateColumn>

<data:DataGridTemplateColumn.HeaderStyle>

<Style TargetType="dataprimitives:DataGridColumnHeader" BasedOn="{StaticResource DataGridColumnHeaderStyle}">

<Setter Property="ContentTemplate">

<Setter.Value>

<DataTemplate>

<TextBlock Text="{Binding Strings.SALES\_Quantity, Source={StaticResource LocalizedStrings}}"/>

</DataTemplate>

</Setter.Value>

</Setter>

</Style>

</data:DataGridTemplateColumn.HeaderStyle>

</data:DataGridTemplateColumn>

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * En-têtes localisées du panier d’achat | **Exemple de démonstration :**   * Soat.SilverlightSalesOrderModule/Views/ShoppingCart/   ShoppingCartView.xaml |

### Binding sur ViewModel dans CellTemplate d’une DataGrid

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Dans mon CellTemplate, je n’arrive pas à binder mon bouton à la commande de mon ViewModel ! | **Solution :**   * RelativeSourceAncestor |

Même constat que pour le paragraphe 5.9.2 (Binding sur ViewModel à l’intérieur d’un DataTemplate), sauf que cela ne fonctionne pas dans une DataGrid ! Espérons que ce bug soit corrigé pour SL4 RTW, car pour l’instant, il ne l’est pas …

En WPF, il est plutôt facile de contourner ce problème en passant par des RelativeSource. Pauvre de nous, Silverlight ne connait que la RelativeSource Self et TemplatedParent. Le TemplatedParent ne fonctionne que dans un ControlTemplate, il s’agit ici d’un DataTemplate.

Dans HappyNet, nous avons dû utiliser une réimplémentation du RelativeSource Ancestor de WPF.

<Button Style="{StaticResource ButtonNoStyle}"

cal:Click.Command="{Binding Tag.RemoveCartItemCommand, RelativeSource={RelativeSource Self}}"

cal:Click.CommandParameter="{Binding}"

Grid.Column="1">

<helper:BindingHelper.Binding>

<helper:BindingProperties SourceProperty="DataContext" RelativeSourceAncestorType="DataGrid"

TargetProperty="Tag"/>

</helper:BindingHelper.Binding>

[...]

</Button>

On a ici une propriété relais qui va chercher la valeur du DataContext (= ViewModel) de la DataGrid parente, et l’assigne à la propriété Tag du Button. Ainsi, il ne reste plus qu’à aller binder la commande du ViewModel défini dans le Tag du bouton.

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Bouton de suppression d’un produit du panier d’achats | **Exemple de démonstration :**   * Soat.SilverlightSalesOrderModule/Views/ShoppingCart/   ShoppingCartView.xaml |

### Changement de l'url (ancre) sans naviguer

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Quand j’essaie de changer l’url, ça me recharge systématiquement la page, alors que je ne veux pas ! | **Solution :**   * Annuler la navigation via l’événément Navigating |

Dans HappyNet, lorsque l’utilisateur clique sur un produit pour voir son détail, l’url change pour garder une trace de cette page. Ainsi, lorsqu’on recharge la page, on retrouve le détail correspondant à l’id passé dans l’url.

Malheureusement, le Navigation Framework ne permet pas de changer d’url à la main, il déclenche automatiquement la navigation, sans crier garde.

On notera que la Frame de Navigation Framework possède un événement Navigating. Lorsque le processus de navigation est lancé, Navigating est déclenché. Heureusement pour nous, il est possible d’annuler le processus grâce à cet event handler !

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Modification de l’url de navigation sans naviguer | **Exemple de démonstration :**   * Soat.SilverlightInfrastructure/Globals.cs * Soat.Silverlight/Shell.xaml.cs |

### Traiter uniquement du dernier appel à un web service

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Quand un utilisateur clique rapidement sur tous mes liens, ça charge les données n’importe comment ! | **Solution :**   * Restreindre au dernier appel aux web services |

Prenons le cas d’HappyNet et de la navigation sur la page des produits. Le comportement de base, sans gestion particulière, est le suivant :



L’utilisateur clique sur une catégorie, appel asynchrone au web service, rapatriement des données, et on affiche.

Prenons maintenant le cas où l’utilisateur s’amuserait à cliquer rapidement sur différentes catégories :



On voit que même en cliquant sur la catégorie « Bikes » en dernier, il se peut que la catégorie affichée soit « Accessories », pour la simple et bonne raison que les produits de type « Accessories » ont mis plus de temps à être renvoyés que les « Bikes ».

De manière à rester fidèle aux actions utilisateurs, et pour éviter des traitements inutiles, il est parfois utile de ne traiter que le dernier appel à un web service.

Pour ce faire, on va utiliser un id unique identifiant l’appel, associé à la magie du UserState (cf.  pour plus de détails), pour ignorer tous les appels sauf le dernier.

|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Affichage des produits d’une catégorie | **Exemple de démonstration :**   * Soat.SilverlightProductionModule/Views/ProductsList/   ProductsListViewModel.cs |

### Texte sélectionnable

|  |  |  |
| --- | --- | --- |
| **http://www.iconarchive.com/icons/custom-icon-design/office/48/info-icon.png** | **Enoncé du problème :**   * + Je ne peux sélectionner aucun texte dans mon application ! Comment je fais mon copié/collé ?! | **Solution :**   * Remplacer les TextBlock par des TextBox en lecture seule |

La réponse est quelque peu simple : plutôt qu’utiliser des TextBlock partout, il convient d’utiliser des TextBox à la place. Pour créer l’illusion de toujours être un TextBlock, il faudra remanier le style pour rendre le fond et les bordures transparents, et mettre la TextBox en lecture seule.



|  |  |
| --- | --- |
| **Fonctionnalité HappyNet :**   * Description du détail d’un produit sélectionnable | **Exemple de démonstration :**   * Soat.SilverlightProductionModule/Views/ProductDetail/   ProductDetailView.xaml |

# Conseils & Bonnes pratiques

* **Snippets / Templates**

HappyNet est livré avec un certain nombre de snippets et templates forts utiles, ne jamais hésiter à en abuser.

* **Astuce du IsHitTestVisible**

Parfois, on ne souhaite pas avoir d’interaction avec certains composants, comme par exemple le clic ou le survol de la souris.

Il est alors utile d’employer l’option disponible sur les UIElement : IsHitTestVisible. Lorsque la valeur est à *false*, on ne peut plus interagir avec l’élément graphique.

Ex. : Prenons un contrôle de loading qui s’animerait au-dessus d’une TextBox. En mettant IsHitTestVisible à *false*, l’utilisateur pourra cliquer « à travers » le loading pour obtenir le focus de la TextBox.

* **MetalScroll : add-in indispensable**

Voilà, c’est dit.



Le principe est simple : la scrollbar est remplacée par une vue type copie d'écran de votre code. On peut ainsi avoir un aperçu rapide des blocs fraîchement modifiés, des breakpoints placés, de commentaires et autres.

Astuce : double-cliquez sur du texte en appuyant sur la touche ALT, et celui-ci se retrouvera surligné dans la scrollbar, rendant la recherche des occurrences tellement plus simple !

[Par ici le téléchargement !](http://code.google.com/p/metalscroll/)

* **On ne touche pas au LayoutRoot !**

Lorsqu’on travaille sur le design, la règle veut qu’on ne touche pas à la Grid LayoutRoot. Donc si vous voulez ajouter des colonnes, ajoutez une Grid en-dessous du LayoutRoot.

Ceci permet de rester flexible si jamais on veut rajouter de nouveaux éléments au design.

* **Répertoire Assemblies contenant toutes les assemblies hors Silverlight SDK**

Valable pour tous vos projets : créer un répertoire contenant toutes les assemblies hors SDK. Il s’agit par exemple du Silverlight Toolkit, de DLL Codeplexiennes, de PRISM, etc.

Lorsque vous travaillez à plusieurs, il est toujours plus agréable de s’assurer que tout le monde tourne sur les mêmes versions, et est capable de compiler sa solution.

* **Splash screen**

Le SplashScreen rend vos applications plus personnalisées et moins amatrices, utilisez-le !

# Composants externes

Parce qu’il est toujours bon de trouver des composants open-sources évitant d’avoir à réinventer la roue, quelques remerciements s’imposent :

**BlackLight** (pour son excellent DragDockPanel) :

<http://blacklight.codeplex.com/>

**ImageTools** (pour la lecture de GIF) :

<http://imagetools.codeplex.com>

**nRoute Toolkit** (framework très intéressant dont j’ai repris quelques composants) :

<http://nroute.codeplex.com/>

**Expression Blend Samples** (Bibliothèque de Triggers, Actions, Behaviors) :

<http://expressionblend.codeplex.com/>

Pensez à aller régulièrement sur [Codeplex](http://www.codeplex.com), c’est une mine de projets en tout genre !