**Création d’une nouvelle vue**

Exemple concret avec un vue utilisée pour le contrôle 4 yeux.

* **Créer un nouveau répertoire et deux sous-répertoires.**

Ils contiendront les classes des vues et vue-modèle pour l’implémentation de la méthode MVVM.

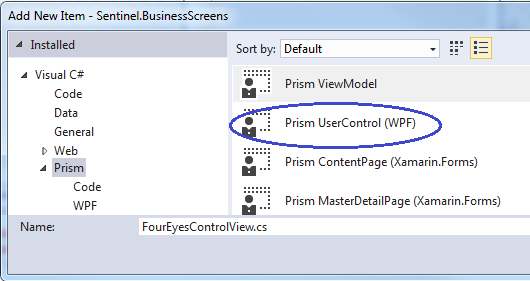
Les noms doivent être pluralisés.



* **Créer la Vue**

Les vues sont des UserControls basés sur le template PRISM.

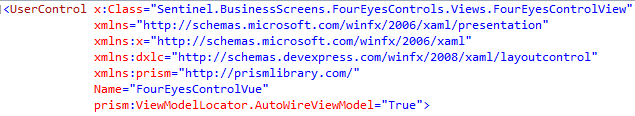
Le nom est au singulier et se termine par « View.xaml »



Dans le XAML de la vue, on ajoute le tag « **Name** » dans le header. Ce nom est le même que celui de la vue en remplaçant « View » par « Vue ». Cela sera potentiellement utile pour l’exécution de scénarios.

On fait également référence à la librairie « layoutControl » de DevExpress (« **dxlc** ») pour le design de base de l’écran.

Le template PRISM rajoute une commande permettant à la vue de se lier automatiquement à son view-model en fonction du nom « prism:ViewModelLocator.AutoWireViewModel="True" »



Une vue se présente toujours de la manière suivante

Il s’agit d’une groupbox Devexpress pour laquelle on redéfinit :

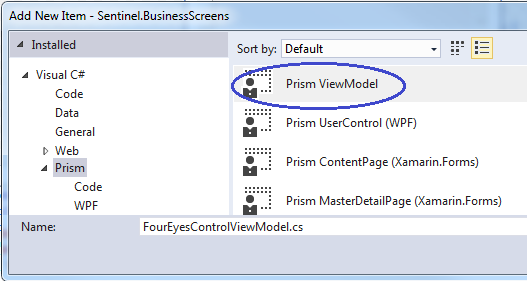
* La couleur de fond.
* La couleur de fond du titre.
* Le header en y ajoutant une image représentative de la fonction de la vue, un libellé et une image cliquable pour fermer la vue.

Et le code correspondant est :

* **Créer le View-Model**

On utilise le template PRISM.

Le nom est au singulier et se termine par « ViewModel.cs »



Le view-model :

* Hérite de la classe « SentinelViewModel » qui se trouve dans le projet « Sentinel.Mvvm ».
* Implémente l’interface « INavigationAware » de « Prism.Regions » permettant de définir des méthodes servant de points d’entrée/sortie de la vue lors de la navigation.



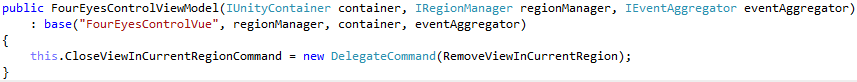
Le constructeur reçoit par injection de dépendance les composants suivants :

« IUnityContainer » de « Microsoft.Practices.Unity »

« IRegionManager » de « Prism.Regions »

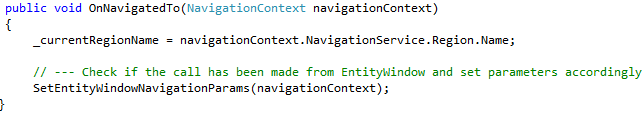
« IEventAggregator » de « Prism.Events »

Ils sont réinjectées dans la classe de base avec le nom de la vue pour la mise en place et l’exécution d’éventuels scénarios.

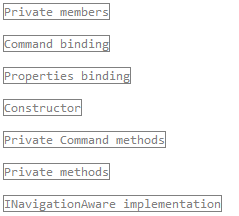


Dans le constructeur, on assigne la commande de fermeture de la vue à une méthode déjà existante (dans la base) ou une méthode personnalisée.

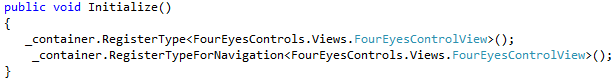
Dans l’implémentation de l’interface « InavigationAware », la méthode « OnNavigatedTo » doit initialiser le nom de la région courante dans laquelle la vue est ouverte ainsi que des propriétés liées à l’ouverture via la fenêtre entité :



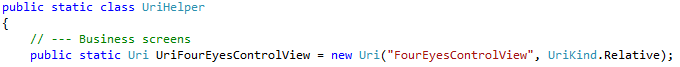
Les différentes régions définies :



On enregistre ensuite la vue dans le conteneur via la méthode « Initialize » de la classe « NomProjetModule.cs »



Enfin, on crée une référence statique à l’Uri de la vue qui sera utilisée lors de la navigation (projet Sentinel.CommonLayer)



* **Créer le point d’entrée et de navigation**

Créer le menu dans la barre de navigation

Constante avec le nom de la tuile et du menu