Program.cs 🡪 Run MainForm

## MainForm : RibbonForm, IMainModule, ISupportViewModel

viewModel es de tipo MainViewModel (o sea que hay una dependencia…)

[1] this.viewModel 🡪 ViewModelSource.Create(() => new MainViewModel(this))

Transforma MainViewModel en una clase MVVM

[3] OnLoad()

Setea SelectedModuleType del view model en ModuleType.Employees.

[6] viewModel\_ModuleAdded

casting del sender como Control, y hace docking en el modulesContainer (el user control que servira para meter las vistas)

Si soporta ribbon, hace el merge de la ribbon que trae el control con la propia del MainForm (tambien de la status bar). De otro modo, hace el Unmerge de la ribbon y la status bar…

[20] OnLoad()

Obtenemos en un array de ModuleTypes[] todos las views que puedo tener en el navigator

Registra la barra de navegacion

StartupProcess 🡪 Succesfully Loaded

## MainViewModel : ViewModelBase, IZoomViewModel

ctor static 🡪 Mvvm.ServiceContainer.Default.RegisterService() registra MessageBoxService, Services.ModuleTypesResolver, Services.ModuleResourceProvider, Services.Win.ModuleResourceProvider

MainViewModel(IMainModule module)

[2] Registra otros servicios ReportActivator, PeekModuleHostingService, etc…

[4] OnSelectedModuleTypeChanged()

SelectedModuleType es una propiedad automatica, por lo tanto ViewModelSource.Create() le agregó property change notifications. Aparentemente…agregó una llamada a OnSelectedModuleTypeChanged(). Este método está escrito, lo unico que hace Create() es incorporar una llamada…

Setea el efecto de transicion entre módulos y obtiene el servicio de resolucion de modulos: resolver = GetService<IModuleTypesResolver>(). Sin embargo GetModule(SelectedModuleType) invoca a GetService<Services.IModuleLocator>().GetModule(type) 🡪 esta en ViewModelBase

Uno de los pasos es SelectedModule=GetModule(SelectedModuleType)

[5] OnSelectedModuleChanged()

Si habia un modulo cargado, se invoca al evento ModuleRemoved()

Se llama al evento SelectedModuleChanged (si hay interesados)

Si el SelectedModule no es null, se verifica que el SelectedModule acepte o no un ParentViewModel, de ser asi se colocaria el ParentViewModel en this (MainViewModel) y aunque aca no se usa tambien se setearian los parametros que se pasan entre view models

Se llama al evento **ModuleAdded**

**Command:** SelectModule(ModuleType)

Simplemente setea la propiedad SelectedModuleType en el argumento. Este comando puede ser bindeado en un navigator

ModuleType 🡪 es un enum que contiene un valor para cada tipo de modulo que tiene la app

CollectionUIViewModel 🡪 un ViewModel que no deriva de nada pero que pasa por ViewModelSource.Create() para alojarse en Employees (View)

## Employees : BaseModuleControl

## DetailForm : RibbonForm, IRibbonModule

Permite mostrar un formulario con un View adentro, lo hace mediante **DetailFormDocumentManagerService**

Puede hacer merge de las ribbon

## EmployeeCollectionViewModel : EmployeeCollectionViewModelBase

OnSelectedEntityChanged()

Este metodo viene derivando desde la primer clase: ReadOnlyCollectionViewModelBase en la cual esta vacia. Sin embargo luego se implementa en CollectionViewModelBase donde se llama a RaiseCanExecute() de Edit, Delete y Save pasando como argumento a cada una SelectedEntity (esta propiedad tambien esta automatica en la clase mas base). Luego se reimplementa en EmployeeCollectionViewModel, llamando a RaiseCanExecute() de ShowMap, PrintProfile y otras…