1 down vote accepted

Create a common interface/class known to both module, called IChildWindowService, register the IChildWindowServe/ChildWindowService in the bootstrapper.

//Highly simplified version

//Can be improved by window reuse, parameter options, stronger eventing

public class ChildWindowService : IChildWindowService

{

public ChildWindowService(IServiceLocator container)

{

\_container = container;

}

public void Show<TViewModel>(TViewModel viewModel = null, Action<TViewModel, bool?> callBack = null) where TViewModel is IViewModel

{

var viewName = typeof(TViewModel).Name.Replace("Model", string.Empty);

// In bootstrapper register all instances of IView or register each view one by one

var view = \_container.GetInstance<IView>(viewName);

viewModel = viewModel ?? \_container.GetInstance<TViewModel>();

view.DataContext = viewModel;

var window = new ChildWindow();

window.Content = view;

var handler = (s,e) => { window.Close(); }

viewModel.RequestClose += handler;

view.Closed += (s,e) => { viewModel.RequestClose -= handler; }

// In silverlight all windows show as Modal, if you are using a third party you can make a decision here

window.Show();

}

}

Create a common CompositePresentationEvent, this event will pass the parameters from point a to point b

public class OpenChildWindowWithParameters : CompositePresentationEvent<ParamEventArgs>{}

The ViewModel in Module A raises the Event. The Controller in Module B registers and reacts to the Event. The Controller in Module B takes the child window service as a dependency. When the event is raised the Controller will create the VM in Module B and pass the parameters to it, from the event, it will also use the Service to display a ChildWindow.