# Замануха

Есть две статьи на Хабре, касающиеся страничной навигации в WPF

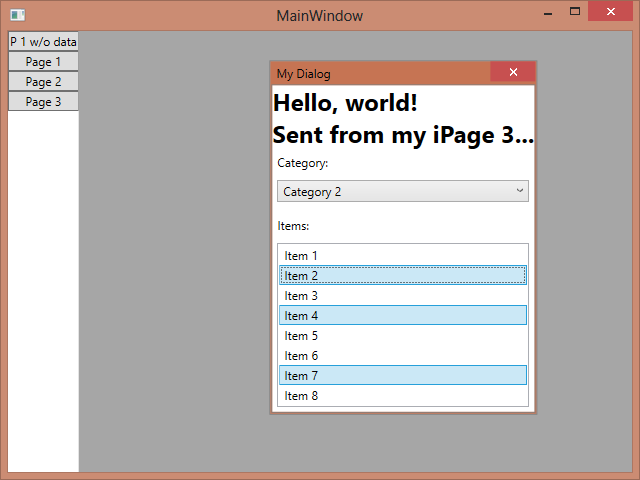
* <http://habrahabr.ru/post/63767/> - Добавление контролов в NavigationWindow
* <http://habrahabr.ru/post/140749/> - Используем связку Page + Frame

В этих статьях рассказывается в основном о создании контролов и совсем немного о реализации самой навигации. Этой статьей хочется восполнить данный пробел и представить готовое решение навигации для WPF приложения на основе паттерна MVVM.

# Введение

Те, кто уже работал с WPF, наверняка знакомы с паттерном MVVM (если нет, то в конце дал ссылки на его описание). Концепция MVVM несложна и как минимум интуитивно должен быть понятным профит от использования данного паттерна. Чтобы паттерн проявил себя во всей красе, нужно как можно меньше логики помещать в Code Behind контролов и ни в коем случае не использовать прямые ссылки на UI внутри ViewModel’ей. Тогда мы получим большой профит в виде возможности тестирования ViewModel’ей отдельно от контролов. Так же хорошей практикой будет сведение к минимуму инстанциирования ViewModel’ей напрямую в контролах. Полбеды, если контрол сам для себя создает ViewModel конкретного типа – в этом случае просто труднее будет подсунуть контролу какую-нибудь тестовую куклу. А вот когда некий родительский контрол занимается созданием ViewModel’ей для разных частных экранов, код обычно превращается в нетестируемую кучу спагетти. Лучше, если созданием ViewModel’ей будут заниматься другие ViewModel’и.

Как это может выглядеть на практике? Давайте представим себе приложение с панелью навигации и несколькими экранами и диалоговыми окнами. Нечто подобное представлено ниже.



Мы можем разглядеть несколько сущностей: главное окно, панель навигации с кнопками, текущая страница и диалог над этой страницей. В нашем приложении для страничной навигации можно было бы использовать HyperLink, подложив вместо кнопок TextBlock с HyperLink в качестве контента. У HyperLink есть свойство, указывающее имя Frame, в котором выполнять переход на новую страницу. И вроде все нормально, можно использовать. Но с использованием HyperLink представляется трудным передача странице нужной ViewModel’и.

Я видел в сети решение, когда в событии Navigated фрейма в главном окне приложения через Code Behind добирались до загруженного во фрейм содержимого и подкидывали туда созданную там же в Code Behind ViewModel. Таким образом, в одном обработчике события придется реализовать длинную портянку if…else if… либо switch. Про то, что тестирование такого Hard Coded процесса навигации крайне трудно автоматизировать, я молчу. Где-то я даже видел создание экземпляра Page и ViewModel’и под нее, подкладывание ViewModel’и в DataContext и вызов Navigate у фрейма с передачей созданного экземпляра Page. Это решение немного лучше предыдущего, но по-прежнему совсем не MVVM-way. Потому код не привожу, чтоб не дай Бог не скопипастили.

Какое решение могло бы более-менее гладко вписаться в контекст MVVM? У класса Page в Silverlight есть перегружаемый метод OnNavigatedTo. В этом методе было бы удобно принимать ViewModel, переданную в NavigationService.Navigate(Uri uri, object navigationContext) вторым параметром. Однако в WPF такого метода нет (ну или я не нашел его или чего-то эквивалентного). Таким образом, нам нужен некий посредник или, если хотите, менеджер, который будет контролировать переходы по страницам и перекладывать из параметра метода в DataContext нужную ViewModel.

О реализации такого менеджера навигации и пойдет речь в данной статье.