**HOW DO I …**

Add Fast Application Switching and Tombstoning support?

In this video we are going to explore Windows Phone Mango’s new application lifecycle model and see how to leverage it in a typical application.

# Overview

Before we begin, let’s have a look at the application that we will be using throughout this video. Wazup is a Silverlight application which allows the user to keep track of several online services: The Windows Phone 7 blog, Digg and Twitter. We will begin by having a quick look at the application and its code.

To get the application to handle deactivation properly we will need to implement tombstoning support, but we should also take advantage of Windows Phone Mango’s fast application switching.

# Fast Application Switching

Fast Application Switching, or FAS, is a new addition to the Windows Phone’s application lifecycle. With FAS, applications stay in memory after deactivation, which makes resuming the application nearly instantaneous, greatly improving the user experience. All you need to do to use FAS in your application is make it target Mango and recompile it.

# Mango Application Lifecycle

To make matters clearer let us review this visualization of the new application lifecycle model.

# Upgrading to Mango

Upgrading the project to Windows Phone Mango is as simple as right clicking the project file and selecting the appropriate option from the context menu.

Note that this is a one way process and projects cannot be downgraded later on.

While the application does not contain any code to support proper tombstoning, it can still benefit from Mango’s new fast application switching.

While FAS apparently makes saving the application’s state redundant, there is no guarantee that the application won’t lose it’s in-memory image eventually, and so we still need to write code for preserving the application’s state:

1. Start with the solution after it has been converted to target Mango.
2. Add **StateManagement.cs** from the “Code” folder to the project. This simply makes saving values to the state dictionary more convenient.
3. Open **BlogPage.xaml.cs** and add the following constants to the **BlogPage** class:

private const string LastPostsKey = "LastPostsKey";

private const string PostsKey = "PostsKey";

private const string CommentsKey = "CommentsKey";

private const string ImagesKey = "ImagesKey";

private const string DisplayedKey = "DisplayedKey";

1. Add a handler to save the page’s state when navigating away from it:

protected override void OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)

{

    this.SaveState(LastPostsKey, LastPosts);

    this.SaveState(PostsKey, Posts);

    this.SaveState(CommentsKey, Comments);

    this.SaveState(ImagesKey, Images);

    this.SaveState(DisplayedKey, PanoramaControl.SelectedIndex);

}

1. Modify the **OnNavigatedTo** handler to restore the page’s state by adding the following lines at the very beginning of the handler:

// try to load data from state object

LastPosts = this.LoadState<ObservableCollection<RssItem>>(LastPostsKey);

Posts = this.LoadState<ObservableCollection<RssItem>>(PostsKey);

Comments = this.LoadState<ObservableCollection<RssItem>>(CommentsKey);

Images = this.LoadState<ObservableCollection<ImageItem>>(ImagesKey);

PanoramaControl.DefaultItem = PanoramaControl.Items[this.LoadState<int>(DisplayedKey)];

1. Open **DiggPage.xaml.cs** and add the following constants to the **DiggPage** class:

private const string SearchTextKey = "SearchTextKey";

private const string LastSearchTextKey = "LastSearchTextKey";

private const string DiggSearchResultsKey = "DiggSearchResultsKey";

1. Add a handler to save the page’s state when navigating away from it:

protected override void OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)

{

    this.SaveState(SearchTextKey, SearchText);

    this.SaveState(LastSearchTextKey, LastSearchText);

    this.SaveState(DiggSearchResultsKey, DiggSearchResults);

}

1. Add a handler to load the page’s state when navigating to the page:

protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)

{

    SearchText = this.LoadState<string>(SearchTextKey);

    LastSearchText = this.LoadState<string>(LastSearchTextKey);

    DiggSearchResults = this.LoadState<ObservableCollection<DiggStory>>(DiggSearchResultsKey);

}

1. Open **TrendsPage.xaml.cs** and add the following constant to the **TrendsPage** class:

private const string TrendsKey = "TrendsKey";

1. Add a handler to save the page’s state when navigating away from it:

protected override void OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)

{

    this.SaveState(TrendsKey, Trends);

}

1. Modify the **OnNavigatedTo** handler to restore the page’s state by adding the following line at the very beginning of the handler:

Trends = this.LoadState<ObservableCollection<Trend>>(TrendsKey);

1. Open **TwitterPage.xaml.cs** and add the following field and constants to the **TwitterPage** class:

private Trend restoredTrend;

private const string TrendsKey = "TrendsKey";

private const string CurrentTrendKey = "CurrentTrendKey";

1. Add a handler to save the page’s state when navigating away from it:

protected override void OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)

{

    this.SaveState(TrendsKey, Trends);

    this.SaveState(CurrentTrendKey, CurrentTrend);

}

1. Modify the **OnNavigatedTo** handler to restore the page’s state by adding the following lines at the end beginning of the handler:

else

{

    // get data from state object

    Trends = this.LoadState<ObservableCollection<Trend>>(TrendsKey);

    restoredTrend = this.LoadState<Trend>(CurrentTrendKey);

}

1. Modify the **PivotControl\_SelectionChanged** to better handle selection change events that are fired when the application is reactivated. Replace the first if condition and its contents with the following:

if (e.AddedItems.Count == 0 || e.AddedItems[0] == null)

{

    return;

}

if (restoredTrend != null)

{

    // When returning from tombstoning, we get a selection change for the first

    // trend which we need to mitigate

    CurrentTrend = restoredTrend;

    restoredTrend = null;

    return;

}

# Summary

In this video, we learned about Windows Phone Mango’s new fast application switching and saw how it integrates into the application lifecycle and enhance the use experience. However, we must remember that fast application switching does not replace proper tombstoning handling.