Published: 8/22/2011

Windows Phone

Windows Phone Fast Application Switching

Rob S. Miles | Microsoft MVP | University of Hull, UK
Andy Wigley | Microsoft MVP | Appa Mundi

Session 5.0

NOKIA

Course Schedule

- Session 1 – Tuesday, August 23, 2011
  - Building Windows Phone Apps with Visual Studio 2010
  - Silverlight on Windows Phone—Introduction
  - Silverlight on Windows Phone—Advanced
  - Using Expression to Build Windows Phone Interfaces
  - **Windows Phone Fast Application Switching**
  - Windows Phone Multi-tasking & Background Tasks
  - Using Windows Phone Resources (Bing Maps, Camera, etc.)
- Session 2 – Wednesday, August 24, 2011
  - Application Data Storage on Windows Phone
  - Using Networks with Windows Phone
  - Tiles & Notifications on Windows Phone
  - XNA for Windows Phone
  - Selling a Windows Phone Application

NOKIA Microsoft

Topics

- The Windows Phone execution model
- Application State Management
- Fast Application Switching
- Dormant programs and Tombstoning
- Application Navigation and Application Switching

NOKIA Microsoft

Published: 8/22/2011

Windows Phone Task Management

- The Windows Phone platform uses a multi-tasking operating system
- This allows phone functions to operate simultaneously
  - You can listen to music and read your email
- However, this multi-tasking ability is not extended to applications that we write
  - Although we can create special applications that can run in the background (more later)

NOKIA

4Microsoft

Published: 8/22/2011

Why Single Task?

- Removing multi-tasking stops one application from being affected by another as it runs
  - Background tasks can impact on performance and battery life
- The limited screen space available to an application makes it hard to manage multiple programs at the same time
- The Windows Phone design does provide for fast task switching by the user
  - Background tasks are special kinds of applications that play a particular role, depending on the needs of the application

NOKIA

5 Microsoft

Published: 8/22/2011

The Back and Start buttons

- The Windows Phone user interface makes use of two buttons, Back and Start
- Pressing Start always opens up the Start menu so that a new application can be selected
- The Back button can be used to exit from one program and return to one being used earlier
- The system maintains a stack of applications to allow navigation in and out of tasks
- Users do not have to restart applications that they want to return to, instead they can just go back to them using the Back button

NOKIA

6 Microsoft

Published: 8/22/2011

The "Long Press" Back Button

- The Back button also has a "long press" behaviour
- This allows the user to select the active application from the stack of interrupted ones
- Screenshots of all the applications in the stack are displayed and the user can select the one that they want
- The application is brought directly into the foreground if it is selected by the user

NOKIA

Microsoft

Published: 8/22/2011

Windows Phone

Demo

Demo 1: Fast Application Switching

The Windows Phone "Back Stack"

- The Windows Phone system maintains a list of applications in the "Navigation Stack" or "Back Stack"
- Each time a new application is started it is put on the top of the "Navigation Stack"
- The user can then return to any of the "Navigation Stack" applications at some time in the future

NOKIA

9 Microsoft

Published: 8/22/2011

Dormant and Tombstoned

- If the user navigates away from an application it becomes *dormant*
  - It is still in memory, along with all its data, but it is not running
  - It may be directly resumed at a later time
- If a dormant application is removed from memory it is said to be *tombstoned*
  - It is no longer in memory, but it might still be resumed because it is still in the *back stack*

NOKIA

10 Microsoft

Software Implications

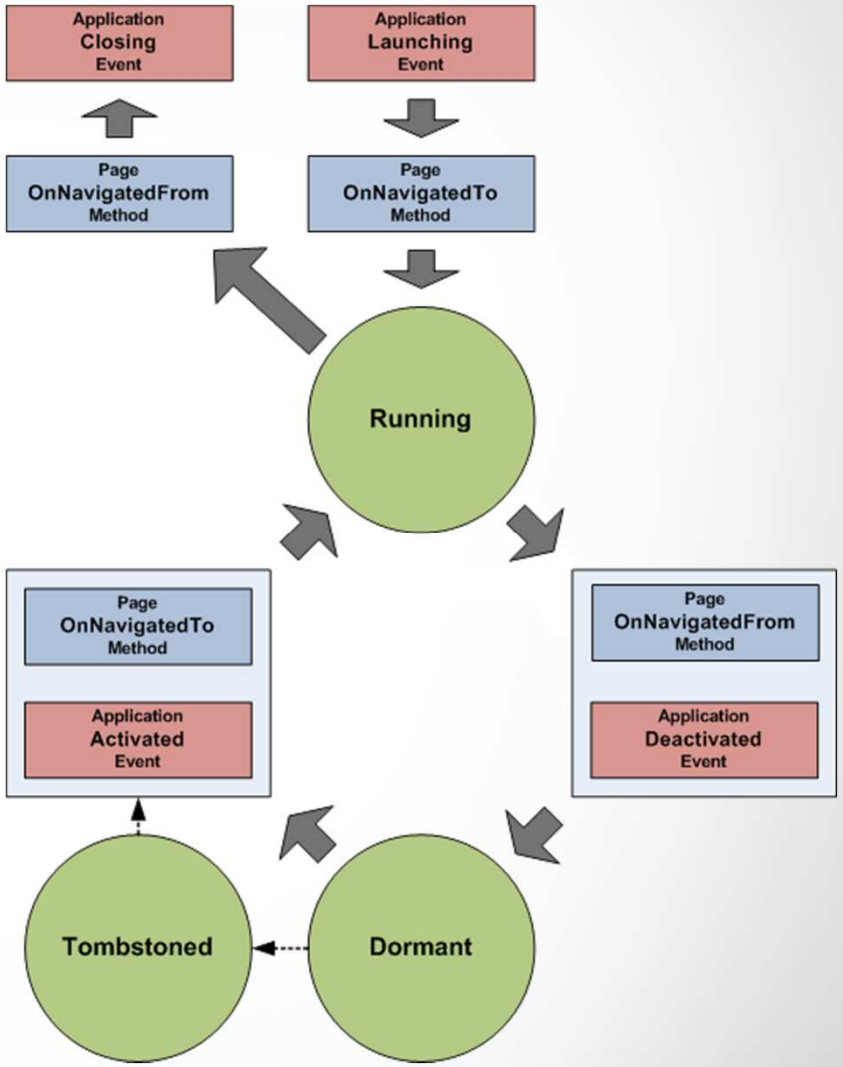
- An application will receive events to notify it of changes to state
- It can also use status information when it resumes
- It has access to memory space to save transient state and isolated storage to persist application state
- An application must use these facilities to give the appearance of one which was never stopped and started
- Resources and settings must be stored and reloaded automatically

NOKIA

11 Microsoft

FAST in action

- This shows the states and the messages that are sent to a Silverlight application when it runs
- We can run through an application lifecycle

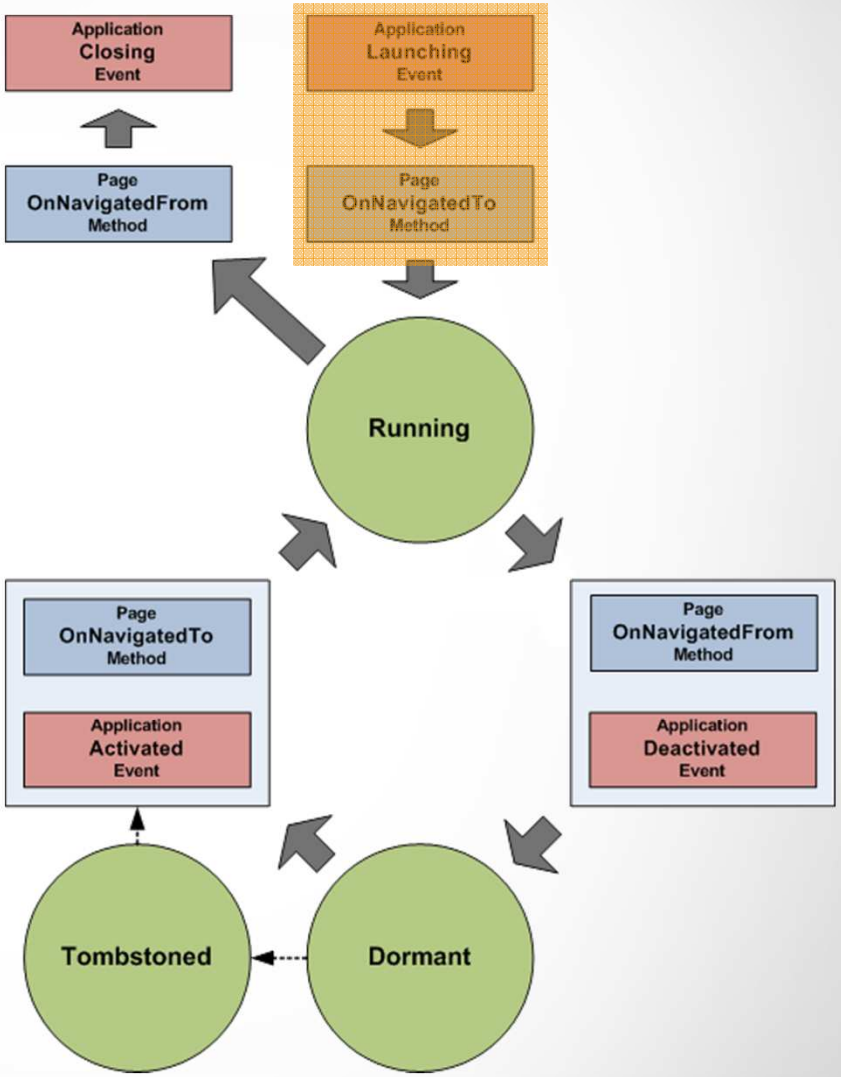


NOKIA

Microsoft

Startup

- The users selects the program and runs it for the very first time
- The application is launched "from cold"
- It moves into the running state in the foreground on the screen
- The "Launching" event is fired on the application
- The "OnNavigatedTo" method is called in the main page of the Silverlight application

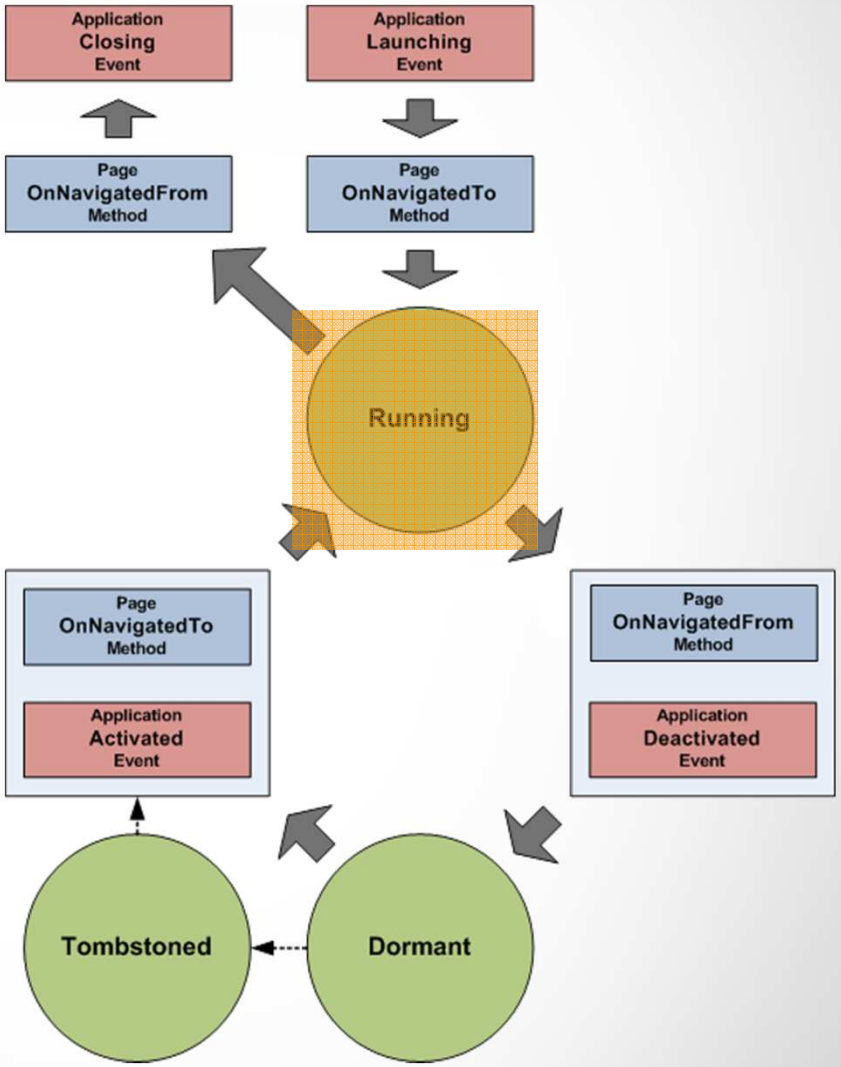


NOKIA

Microsoft

Running

- The user interacts with the application
  - Pressing buttons
  - Entering data
  - Viewing program responses
- All threads, web requests and other program activities are active and the program can make as many demands on the system as it likes

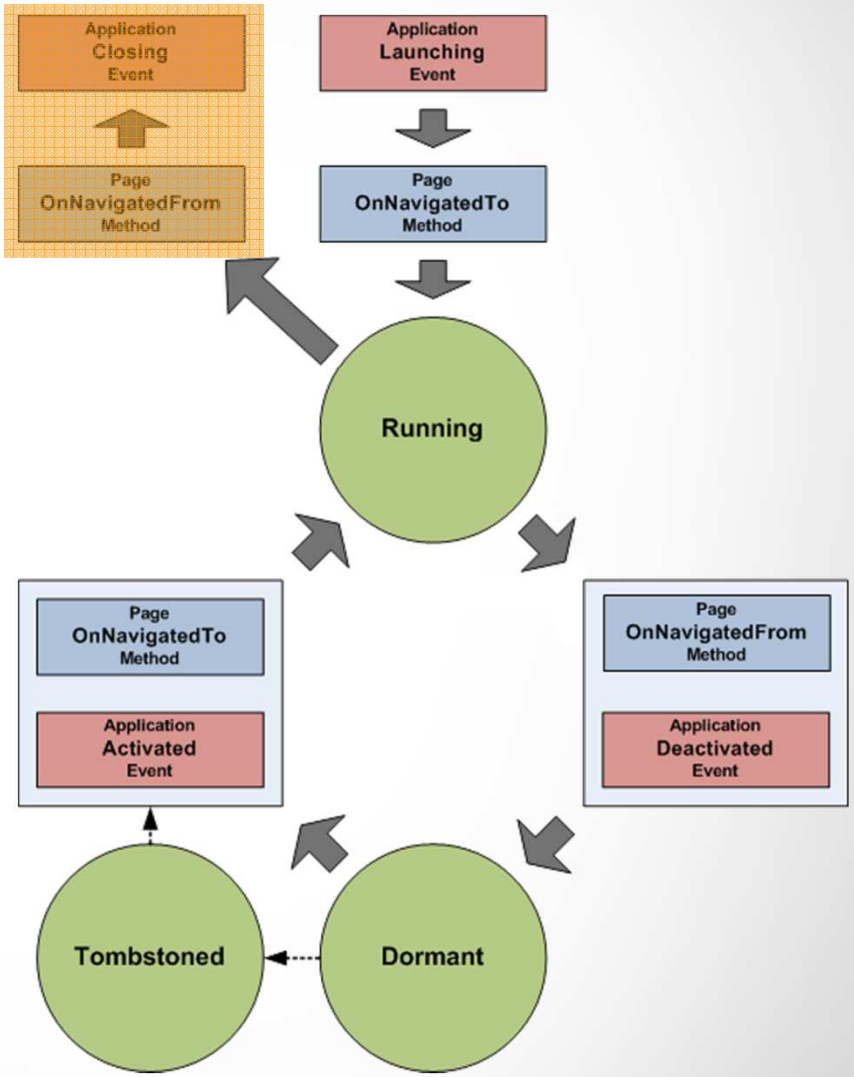


NOKIA

Microsoft

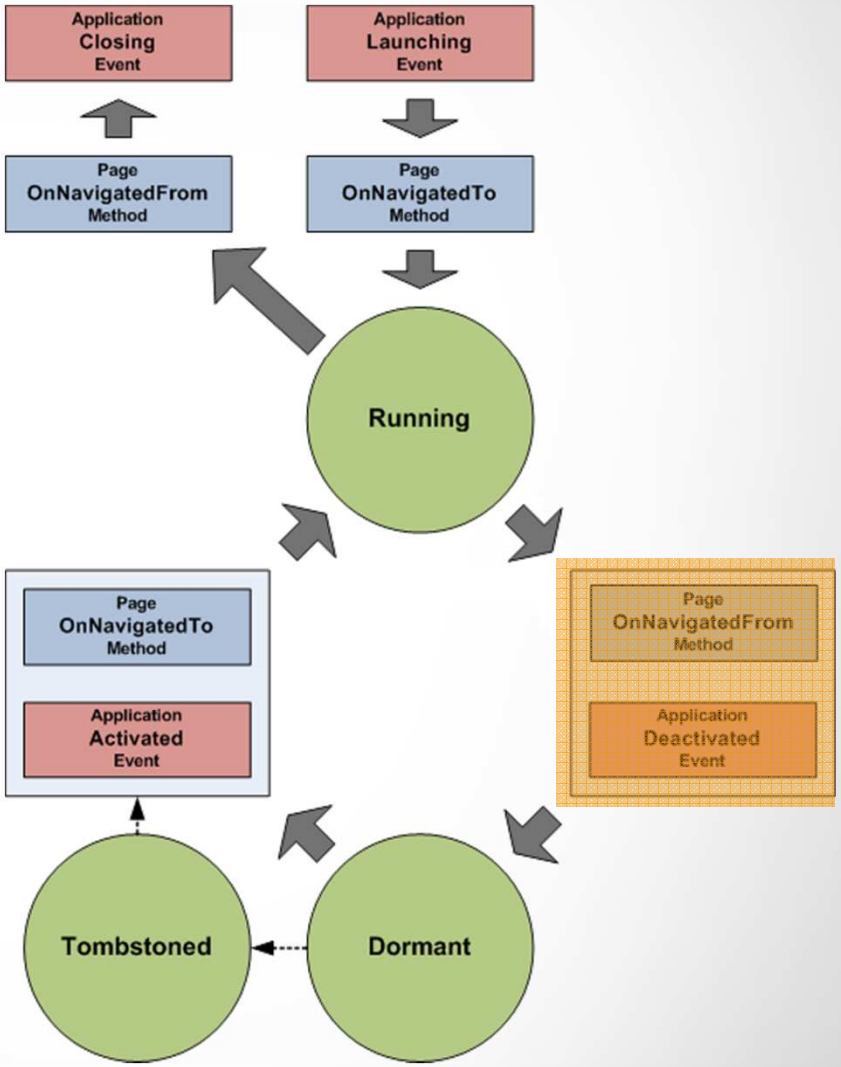
Application Closed

- The user can close the application by pressing "Back" at the top menu of it
  - The OnPageNavigatedFrom method is called to tell the page the user is leaving it
  - The Closing event is called to give the application a chance to save data
- Your application will be halted and removed from memory
  - You can override the Back button behaviour if you want to get control before your application is closed



Navigate from

- The user does something that moves away from your app
  - Selects another application
  - Presses Start
  - Presses Search
- The application is made Dormant
  - OnNavigatedFrom method is called
  - Application Deactivated event is fired

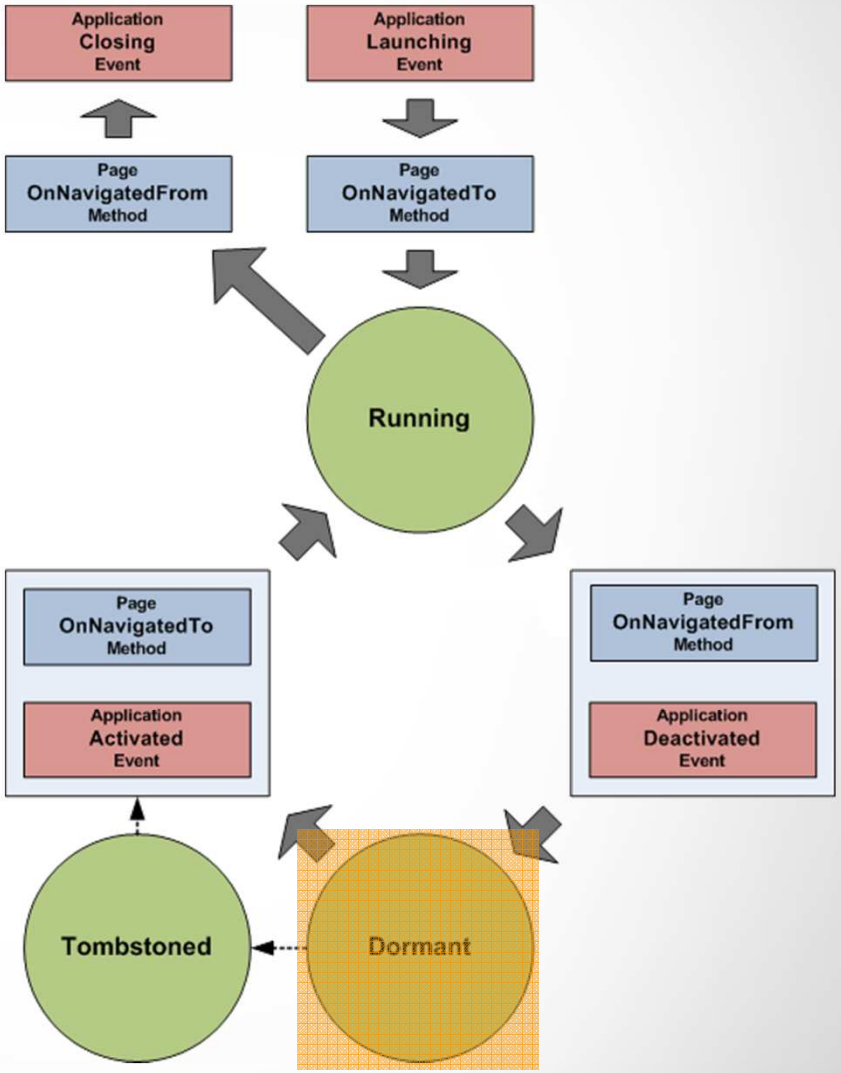


NOKIA

Microsoft

Dormant

- When an application is dormant all its program code and data are still in memory
- However, nothing is running
  - All threads are stopped
  - No response from any asynchronous requests
- Waking an application from dormant is very easy to do and the application does not have to reconstitute any data

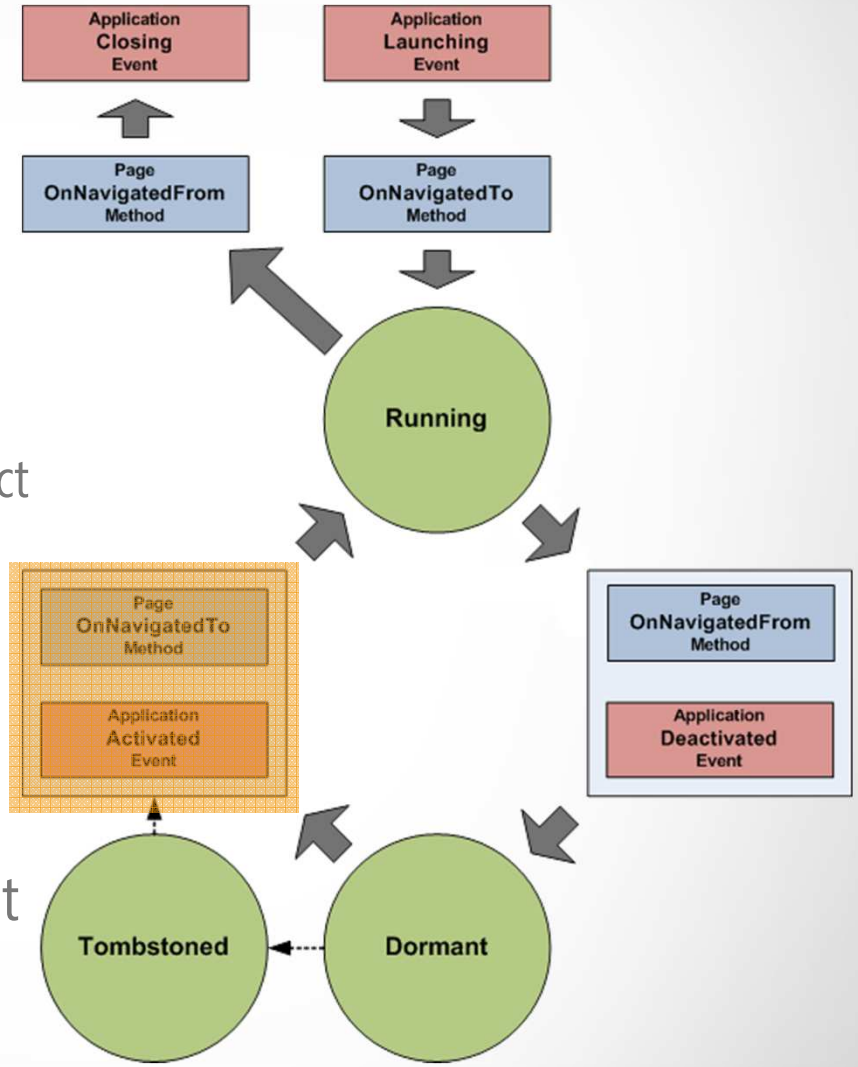


NOKIA

17 Microsoft

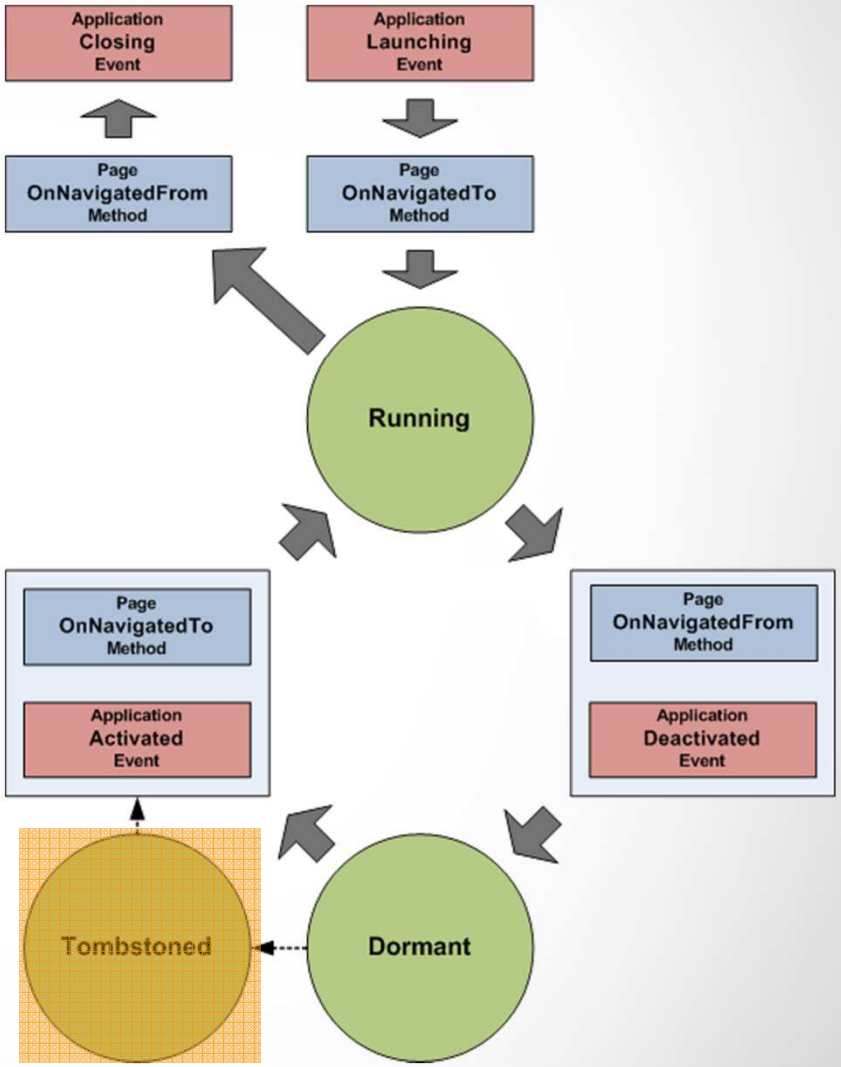
Resumed

- The user returns to the dormant application
  - Pressing Back to exit an application or using the "Long Press" on back to select the application from those running
- Activated event is fired on the application and the OnNavigatedTo method is called in the active Silverlight page



Tombstoned

- If an application is "tombstoned" it is removed from memory
- This happens when there is not room in memory for all the dormant applications
- Your application does not get a message when it is moved from the Dormant to the Tombstoned state
- Applications will be tombstoned when memory becomes scarce

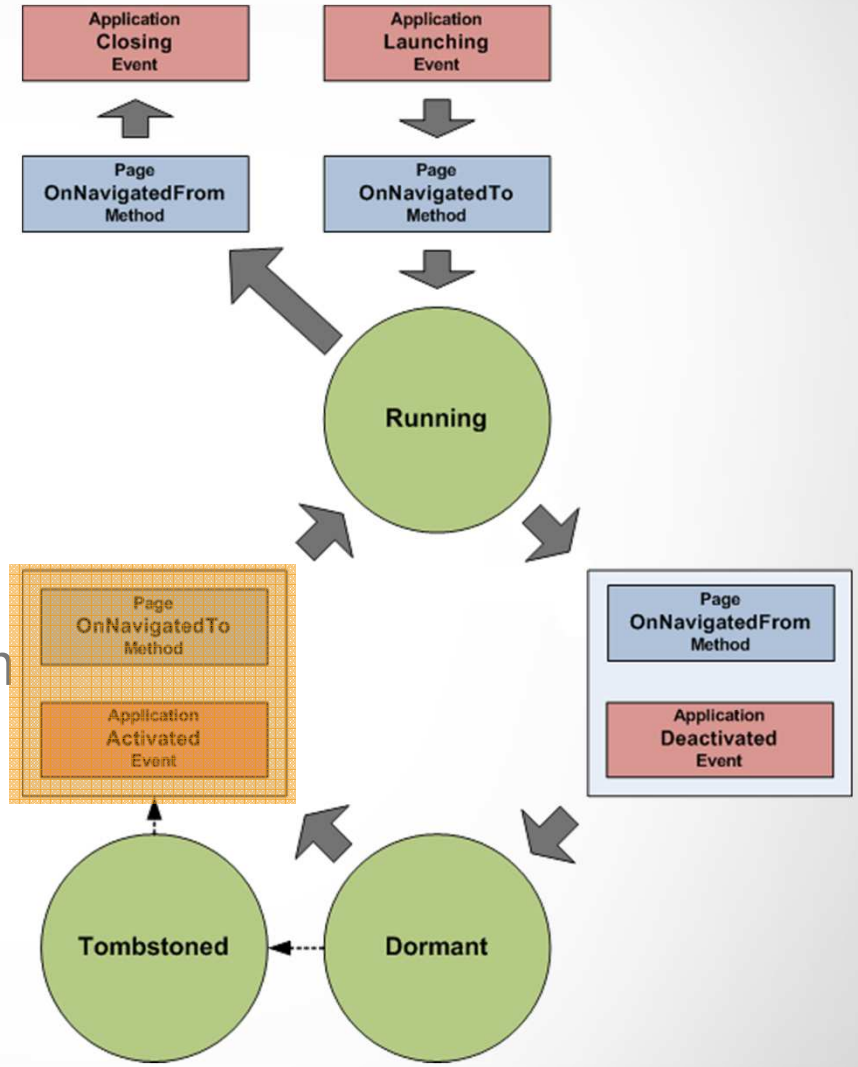


NOKIA

Microsoft

Resuming

- The messages your application gets when resumed are **the same** whether it is being resumed from Dormant or Tombstoned state
- However, the application can determine the mode of the resume when it starts up
  - It can choose not to reload data if it is being resumed from the Dormant state



NOKIA

Microsoft

Published: 8/22/2011

Finding the Resume type

```
private void Application_Activated(object sender, ActivatedEventArgs e)
{
    if (e.IsApplicationInstancePreserved)
    {
        // Dormant - objects in memory intact
    }
    else
    {
        // Tombstoned - need to reload
    }
}
```

- The Activation handler can test a flag to determine the type of resume taking place

NOKIA Microsoft

Captain's Log

- This is a simple Silverlight application that can be used to keep a log
- Entries can be created and stored in the log
- Also an option to view all the log entries on a second page
- The program should persist all the data when the user navigates away from it

NOKIA

22 Microsoft

Data in the Captain's Log

```
// This is shared amongst all the pages
// It is the contents of the log itself, as a large string
public string LogText;

// This is also shared, only used by the log entry page
public string LogEntry;
```

- These are the two data items that the log uses
  - A string which forms a partially completed log entry
  - A string that holds all the log entries that have been entered
- These are held in the application
  - When a page is navigated to it will load the text into it

NOKIA Microsoft

Published: 8/22/2011

Displaying the Data

```
protected override void OnNavigatedTo(
                    System.Windows.Navigation.NavigationEventArgs e)
{
    base.OnNavigatedTo(e);

    // Get a reference to the parent application
    App thisApp = App.Current as App;

    // Put the text onto the display
    logTextBox.Text = thisApp.LogEntry;
}
```

- When the user navigates to a data page it gets the data from the application and then displays it in the textbox

NOKIA Microsoft

Published: 8/22/2011

Storing the Data

```
protected override void OnNavigatedFrom(
                    System.Windows.Navigation.NavigationEventArgs e)
{
    base.OnNavigatedFrom(e);

    // Get a reference to the parent application
    App thisApp = App.Current as App;

    // Store the text in the application
    thisApp.LogEntry = logTextBox.Text;
}
```

- The reverse behaviour which stores data when the page is left
- This method is called when the application is deactivated

NOKIA Microsoft

Published: 8/22/2011

Windows Phone

Demo

Demo1: Log with no storage

Published: 8/22/2011

A Well Behaved Captain's Log

- We are going to make the Captains Log program "well behaved"
- The program will automatically persist data when the user exits and load on entry
- It will also store data when it is stops running
  - It will store the log and text entered in IsloatedStorage to retain the data when the phone is switched off
  - It will store the log and text entered into the application state object to allow for a quick return if the user switches between applications

NOKIA Microsoft

Isolated Storage vs. State Storage

- Isolated storage is so called because the data for an application is isolated from all other applications
  - Can be used as filestore where applications can store folders and files
  - It is slow to access, since it is based on NVRAM technology
  - It can also be used to store name/value pairs, e.g. program settings
- State storage is so called because it is used to hold the state of an application
  - It can be used to store name/value pairs which are held in memory for dormant or tombstoned applications
  - It provides very quick access to data

NOKIA Microsoft

Storage in the Captain's Log

```
private void SaveToIsolatedStorage()
{
    saveTextToIsolatedStorage("Log", LogText);
    saveTextToIsolatedStorage("Entry", LogEntry);
}
```

- There are a set of methods that are called to save and load using either the isolated storage or the state object
- In either case the methods allow the application to store a named string
- This is the method that is called to store the state of the program in isolated storage
- The other three are named using the same conventions

NOKIA Microsoft

Application Switching Events

- There are four events that our application must deal with
  - **Launching** – a new copy of the program has been started by the user
  - **Closing** - the program is ending normally
  - **Deactivated** – the user has moved to another program (e.g. pressed Start)
  - **Activated** – the user is returning to the program (e.g. used "Long Back" press)

NOKIA

30 Microsoft

Switching Event Methods

- Each of the events is mapped onto a method in the `App.xaml.cs` file
  - This is also where the data shared by the forms in our application is shared
- By adding code into these methods an application can get control when the event occurs
- The code in the method can load or save data into the program as appropriate
- We need to populate the methods with correctly behaved code

NOKIA

31 Microsoft

Application Switching Methods

```
private void Application_Launching(object sender,
                                   LaunchingEventArgs e)
{
}
private void Application_Activated(object sender,
                                   ActivatedEventArgs e)
{
}
private void Application_Deactivated(object sender,
                                     DeactivatedEventArgs e)
{
}
private void Application_Closing(object sender,
                                 ClosingEventArgs e)
{
}
```

NOKIA Microsoft

Saving on Closing

```
// Code to execute when the application is closing (eg, user hit Back)
// This code will not execute when the application is deactivated
private void Application_Closing(object sender, ClosingEventArgs e)
{
    // No need to store to the state object - we are not coming back
    SaveToIsolatedStorage();
}
```

- This method is called when the application is being closed
- The application must store the active data into isolated storage
  - This will persist even when the phone is turned off
- The application can then load the data when it is restarted

NOKIA Microsoft

Loading on Launching

```
private void Application_Launching(object sender, LaunchingEventArgs e)
{
    // Note: This is not advisable. For larger amounts of data
    // the application should either start up a thread to perform the load
    // or set a flag to trigger the load later on
    // The Launching method should complete as quickly as possible
    LoadFromIsolatedStorage();
}
```

- This method is called when the application is being launched
- It can load the data for the application from isolated storage
- However, it is important that the method completes quickly, as the program will not be displayed until it has completed

NOKIA Microsoft

Published: 8/22/2011

Windows Phone

Demo

Demo2: Log with storage

Published: 8/22/2011

Saving on Deactivation

```
// Code to execute when the application is deactivated (sent to background)
// This code will not execute when the application is closing
private void Application_Deactivated(object sender, DeactivatedEventArgs e)
{
    SaveToIsolatedStorage();
    SaveToStateObject();
}
```

- The Deactivated method is called when another application replaces ours in the foreground
- The method must save the program data to the state object and to isolated storage
- We must save to isolated storage in case we are never resumed

NOKIA Microsoft

Loading on Activation

```
private void Application_Activated(object sender, ActivatedEventArgs e)
{
    if (!e.IsApplicationInstancePreserved)
    {
        // We are coming back from a tombstone -
        // need to restore from the state object
        LoadFromStateObject();
    }
}
```

- When the application method runs it checks to see if the application is resuming from the Dormant state (which means it has not left memory)
- If this is not the case the application must reload the state data
- It doesn't need to use the isolated storage, it can get this from the state store

NOKIA Microsoft

Published: 8/22/2011

Windows Phone

Demo

Demo3: Fully Working Log

Form Navigation

- The Windows Phone system will remember the position in the application for us
- This means that when the user returns to the application it will automatically go back to the page that was left
- However, it is up to us to make sure that the content of the page is correct

NOKIA Microsoft

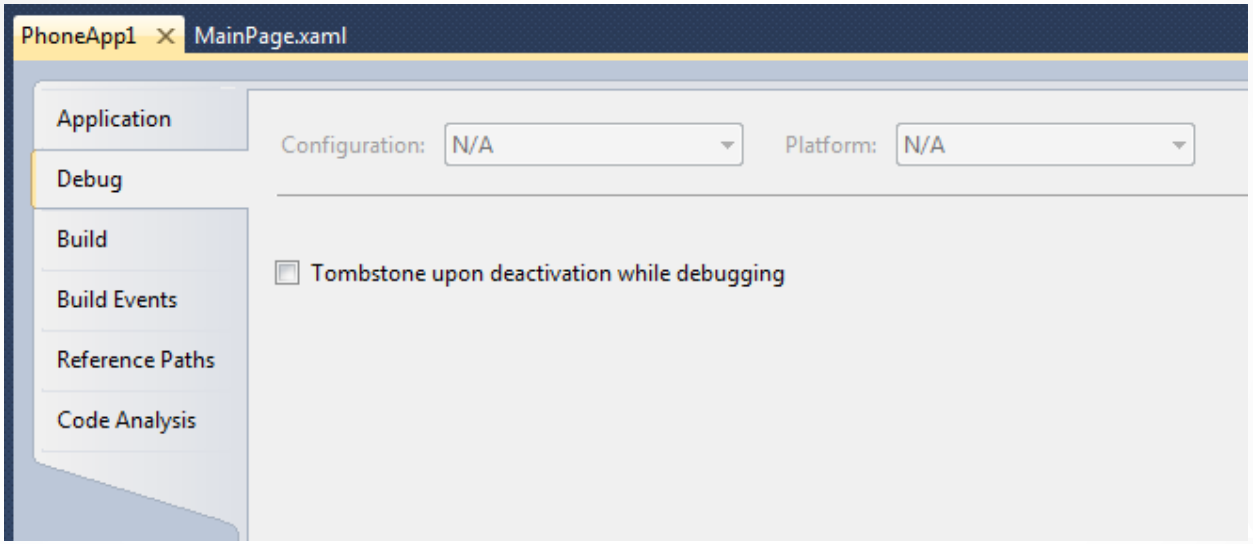
Published: 8/22/2011

Windows Phone

Demo

Demo4: Form Navigation

Forcing Tombstoning



- To force the Windows Phone system to remove your application from memory you should check this box
- It is important that you test this behaviour before submitting the application

NOKIA

41 Microsoft

Review

- Only one Windows Phone application in front of user at any time
- Start and Back buttons on the phone are used to start new applications and return to previously used ones
- If an application is replaced by another it is either made Dormant (still in memory but not running) or Tombstoned (removed from memory)
- Applications must use populate methods provided in the App.xaml.cs class to save and retrieve state information when appropriate
  - State can be stored in memory for quick reload and in isolated storage which serve as a permanent store

NOKIA

42 Microsoft

Windows® Phone

The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation.

MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

Microsoft