



Module

App Lifecycle

Table of contents

- Introduction
- Launching
- Activated App
- Suspending
- Resumed App
- Closing
- Termination
- SuspensionManager
- NavigationHelper

Table of contents

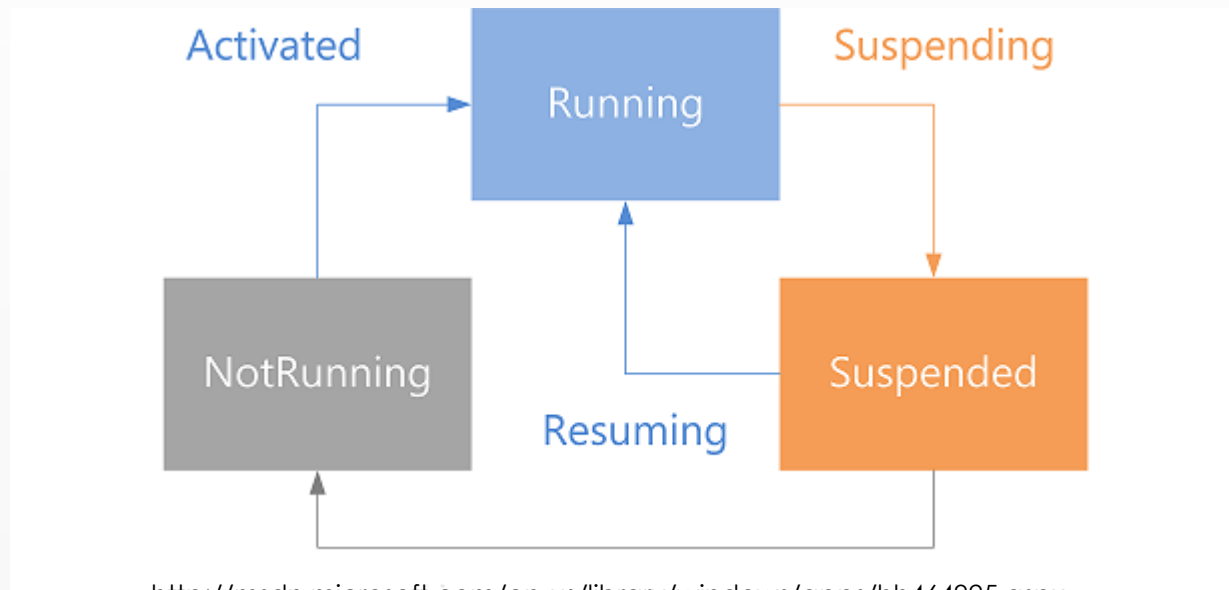
- App Data and Session Data
- App Data
- Session Data
- App Data Saving
- Session Data Saving
- App Data and Session Data Restoring
- Debugger

Introduction

- ▶ The OS runs one foreground app at a time
 - ▶ All other apps are suspended (still in memory) or terminated
- ▶ At one point, an app is
 - ▶ Running
 - ▶ Suspended
 - ▶ Or not running
- ▶ The fact of passing from the 'not running' state to the 'running' state is called activation

Introduction

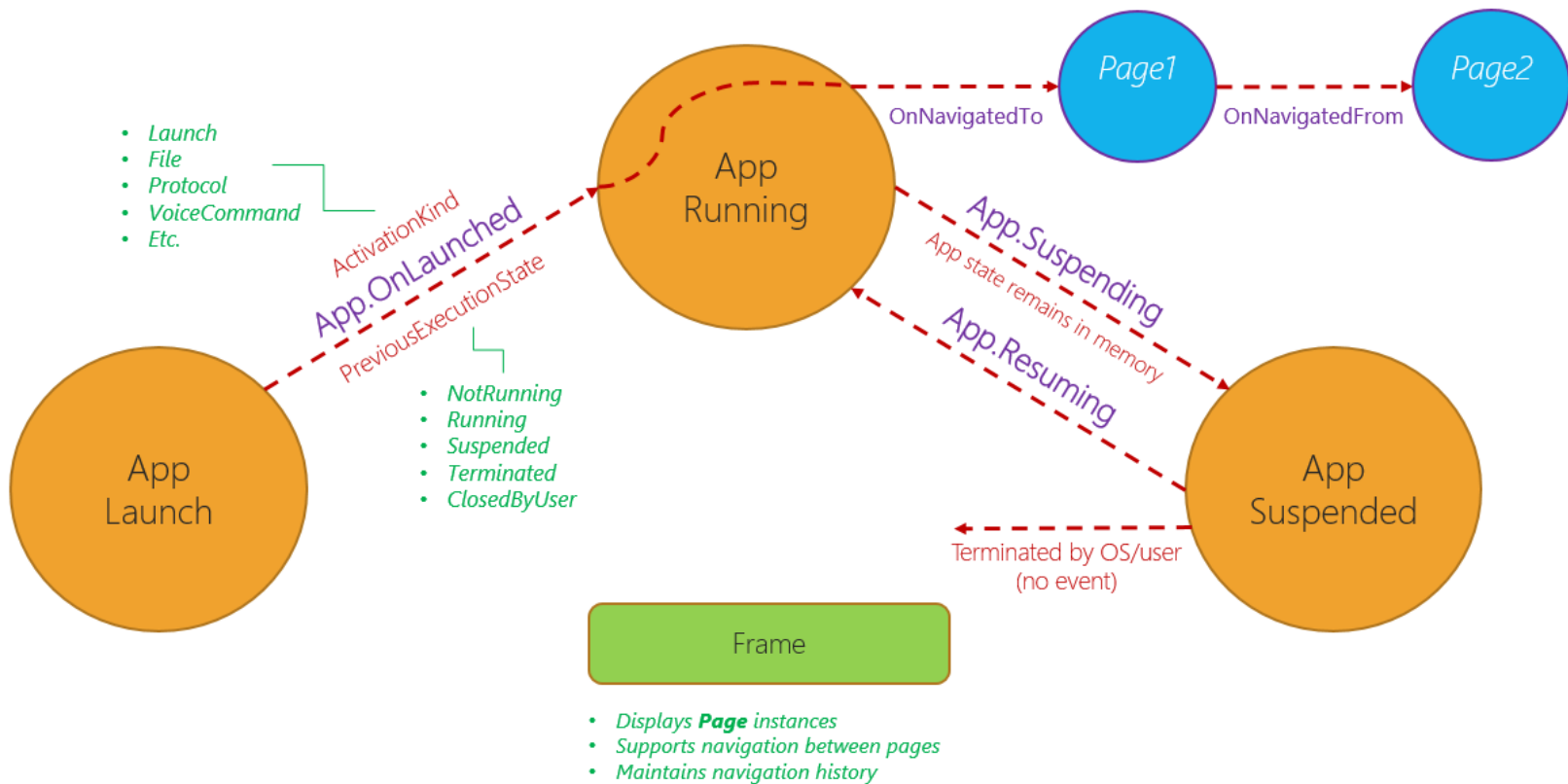
- ▶ An app can be suspended when
 - ▶ The user leaves it for a moment to another one
 - ▶ When the OS detects a state of low supply



<http://msdn.microsoft.com/en-us/library/windows/apps/hh464925.aspx>

Introduction

Basic App Lifecycle



Introduction

- ▶ While an app is suspended, it's still in memory
 - ▶ So, the user can quickly return to the app
 - ▶ No code is required to manage the transition between the suspension and the resumption
- ▶ But the OS can close a suspended app at any time to free up memory or save energy!
- ▶ When the user closes an app with Alt-F4 or with another movement, the app is suspended during 10'' before being closed

Introduction

- The OS warns the app before suspending, not before closing
- Events to manage
 - OnSuspending
 - To store the state
 - To free resources and file descriptors
 - OnLaunched
 - To restore the state
 - To give the impression to the user that the application has never stopped running

Introduction

- Events to manage (...)
 - OnActivated
 - When a user launches the app normally (for example, by tapping the app tile), only the **OnLaunched** method is called
 - Method Invoked when the application is activated by some means other than normal launching
 - For example, from another app through the Search contract
 - The programmer must override the OnActivated method

Introduction

- ▶ Events to manage (...)
 - ▶ OnResuming
 - ▶ The system suspends the app whenever the user switches to another app
 - ▶ The system resumes the app whenever the user switches back to it
 - ▶ The content of the variables and data structures is the same as it was before the system suspended the app
 - ▶ The system restores the app exactly, so that it appears to the user as if it's been running in the background

Introduction

- ▶ Events to manage (...)
 - ▶ OnResuming (...)
 - ▶ However, the app may have been suspended for a significant amount of time
 - ▶ So any displayed content that might have changed while the app was suspended should be refreshed
 - ▶ E.g., news feeds or the user's location
 - ▶ If the app doesn't have any displayed content to refresh, there's no need to handle the **Resuming** event

Introduction

- When occurs each state transition?
- What should do the app in response?
- Enumeration
 - *ApplicationExecutionState*

<i>ApplicationExecutionState</i>	
NotRunning	The app is not running
Running	The app is running
Suspended	The app is suspended
Terminated	The app was terminated after being suspended
ClosedByUser	The app was closed by the user

Launching

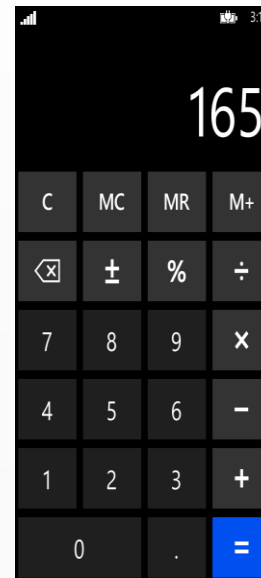
- ▶ An app is launched
 - ▶ When it is activated by the user and the app process was previously in the **NotRunning** state
- ▶ An app could be in the **NotRunning** state because
 - ▶ It has never been launched
 - ▶ It was running but then crashed
 - ▶ It was suspended but then couldn't be kept in memory and was terminated by the system

Launching

➡ E.g.,



Not Running



Running

Launching

- ▶ When an app is launched
 - ▶ The OS displays a splash screen for the app
- ▶ Splash screen
 - ▶ Usually an image that appears to the user to wait while the application is preparing to be launched
 - ▶ By default, implemented in the folder *Assets*
 - ▶ `SplashScreen.scale-xxx.png`
 - ▶ Can be changed by the programmer

Launching

- ▶ While the splash screen is displayed
 - ▶ An app should ensure that it's ready for its user interface to be displayed to the user
 - ▶ The primary tasks for the app are to register event handlers and set up any custom UI it needs for loading
 - ▶ These tasks should only take a few seconds



If an app needs to request data from the network or needs to retrieve large amounts of data from disk, these activities should be completed outside of activation

Launching

- If the user launches the app after the OS has terminated it,
 - The app receives an *Activated* event
 - The user sees the splash screen of the app until the app is activated
- After the app completes activation
 - The state of the App is *Running*
 - The splash screen disappears

Launching

- In App.xaml.cs : OnLaunched method

Activated event argument

```
public async protected override void OnLaunched(LaunchActivatedEventArgs e)
{
    if (e.PreviousExecutionState == ApplicationExecutionState.Terminated ||
        e.PreviousExecutionState == ApplicationExecutionState.ClosedByUser)
    {
        // Populate UI with the previously saved application data
    }
    else
    {
        // Populate UI with defaults
    }

    EnsurePageCreatedAndActivate();
}
```

State of the app before it was activated

Activated App

- ▶ An app can be activated by the user through a variety of contracts and extensions
- ▶ To participate in activation, the app must register to override the **OnActivated** method
 - ▶ App's activation event handler can test to see
 - ▶ Why it was activated
 - ▶ Whether it was already in the *Running* state

Activated App

- ▶ An app can use activation
 - ▶ To restore the data previously saved in the event when the OS ends the app and if the user re-launches it
 - ▶ To restore the data previously saved when it was last suspended
 - ▶ Whether the app's default data must be loaded


Suspending

- ▶ An app can be suspended when
 - ▶ The device enters in a low power state
 - ▶ The user switches away from it
 - ▶ The app is moved to the background
 - ▶ The OS waits a few seconds to see whether the user immediately switches back to the app
 - ▶ If the user does not switch back, the OS suspends the app
 - ▶ If the user switches back to it before it can be suspended, the app remains in the *Running* state

Suspending

- Windows can terminate a suspended app
 - To free up memory for other apps
 - To save power
- When the app is terminated
 - It stops running
 - It is unloaded from memory
- When the user closes an app (Alt+F4 or Close gesture)
 - The app is suspended for 10 seconds and then terminated

Suspending

- Windows notifies the App when the OS suspends it
- But Windows doesn't provide additional notification when it terminates the app
- The App must handle the *Suspended* event in order to look like it never stopped running
- The App
 - Needs to retain any data the user has entered, settings he has changed, ...
 - Must save its state when it's suspended
 - In case Windows terminates it, its state could be restored later

Suspending

- In App.xaml.cs: OnSuspending method

```
/// <summary>
/// Invoked when application execution is being suspended. Application state is saved
/// without knowing whether the application will be terminated or resumed with the contents
/// of memory still intact.
/// </summary>
/// <param name="sender">The source of the suspend request.</param>
/// <param name="e">Details about the suspend request.</param>
1reference
private void OnSuspending(object sender, SuspendingEventArgs e)
{
    var deferral = e.SuspendingOperation.GetDeferral();

    // TODO: Save application state and stop any background activity
    deferral.Complete();
}
```

This code has limited time to run.
Do the minimum amount of work!

Resumed App

- ▶ A suspended app is resumed
 - ▶ When the user switches to it
 - ▶ When the device comes out of a low power state
- ▶ When an app is resumed from being suspended
 - ▶ The same App process is resumed
 - ▶ The app's memory state is preserved, variable values are retained!

Resumed App

➡ In the App.xaml.cs,

```
public App()
{
    this.InitializeComponent();
    this.Suspending += OnSuspending;
    this.Resuming += OnResuming; // to add
}

private void OnResuming(object sender, object e)
{
    // TODO: whatever you need to do to resume your app
}
```

Closing

- Users don't need to close apps
 - It's the role of the OS
 - The developer can't include any UI in the app to enable the user to close the app
- Users can choose to close an app
 - Using the close gesture
 - By pressing Alt+F4 on Windows
 - By using the task switcher on Windows Phone

Closing

- No special event to indicate that the user has closed an app
- After an app has been closed by the user
 - It is suspended and terminated
 - *NotRunning* state about 10 seconds

Termination

- ▶ OS can terminate an app
 - ▶ If the user goes to another app and the memory is insufficient
- ▶ The user will return to the first app...
 - ▶ The user will be unaware that the app was terminated
 - ▶ The developer must maintain the illusion that the app has always been running, even though it hasn't

Termination

- The user will return to the first app...
 - The user must not lose data and should continue recent activity seamlessly
 - The app may need to restore transient session state including
 - The page navigation history
 - The page position
 - The parameters
 - The contents of input fields

SuspensionManager

- Class

- To simplify the lifecycle App management
- To save and restore the navigation state of the Frame that hosts the app pages
 - Using an App's *Frame* object
 - Two App-level lifecycle events *OnLaunched* and *OnSuspending* to provide a repository where the state data can be saved (loaded)
- Allows every page to register and to restore its state
 - SuspensionManager serializes the data of state of page
 - It writes them in a XML file in the local storage of the app

SuspensionManager

- In the Common folder in the *Basic* Page Template
- When the developer adds a *Basic* page, or other non-*Blank* type of page to a WinRT project
 - Some common support classes are added
 - Including *SuspensionManager* and *NavigationHelper*
[Delete the *MainPage.xaml*, adjust *App.xaml.cs*]
 - Useful state management mechanism is accessible
 - The piece missing at the moment is a repository to use for holding state
 - The *SuspensionManager* class comes in handy

NavigationHelper

- ▶ Class
 - ▶ Using of *SuspensionManager* to store page-level state data
 - ▶ Simplification "forward/back" navigation mechanism

App Data and Session Data

- ▶ An app always has to seem to have been executed in the eyes of the user
 - ▶ The app has to keep the data which the user entered, the parameters which he modified, ...
 - ▶ The state of the app must be thus recorded as it is suspended, in case Windows would stop it, to allow the later restoration of the state
- ▶ Two types of data can be managed in an app
 - ▶ The app data
 - ▶ The session data

App Data

- The app data
 - Are persistent from a session to another one
 - Must always be accessible to the user
 - E.g., nameStudent is a TextBox that the user is in the process of introducing
- The app has only 5" to execute the code of the administrator of events of suspension
 - The developer has to make so that the important data are registered in a storage persisting before its suspension

Session Data

- ▶ The session data
 - ▶ Are temporary data which are relative to the active session of the user within the app
 - ▶ A session ends when the user
 - ▶ Closes the application (Closure gesture or Alt+F4)
 - ▶ Restarts the computer
 - ▶ Or closes the session on the computer
 - ▶ A session can be restored only if the OS suspend the app and close it
 - ▶ A suspended application is not erased by the memory; no state is lost in this case

SuspensionManager using

➤ Example

```
<!--TODO: Content should be placed within the following grid-->
<Grid Grid.Row="1" x:Name="ContentRoot" Margin="19,9.5,19,0">
    <StackPanel Margin="24,20,20,20">
        <TextBlock FontSize="16" Text="What's your name? " />
        <TextBox Name="nameInput" Width="300" HorizontalAlignment="Left" TextChanged="nameInput_TextChanged"/>
        <Button Content="Say "Salut"" Click="Button_Click"></Button>
        <TextBlock Name="messageOutput" FontSize="16" />
    </StackPanel>
</Grid>
```

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    messageOutput.Text = "Salut, " + nameInput.Text;
}
```

SuspensionManager Using

- **Step 1**: Use SuspensionManager in the App
 - At first, register the main Frame of the App
 - In the OnLaunched method of App.xaml.cs
 - Thus, SuspensionManager
 - Knows every page of the app
 - Can register and restore the state of navigation

```
if (rootFrame == null)
{
    rootFrame = new Frame();
    rootFrame.CacheSize = 1;

    SuspensionManager.RegisterFrame(rootFrame, "appFrame");
}
```

App Data Saving

- **Step 2** : Register the App data

- Solution 1 : Create one *Windows.Storage.ApplicationData* object to manage the App data
 - It has a *RoamingSettings* property which returns one *ApplicationDataContainer* object
 - To store the App data which persists from a session to another one
 - To facilitate the storage of data in a accessible way to the user between several devices

App Data Saving

- ▶ Solution 2 : Create one *Windows.Storage.ApplicationData* object to manage the App data
 - ▶ It has a *LocalSettings* property which returns one *ApplicationDataContainer* object
 - ▶ Local App data container for local parameters
 - ▶ Used only to store information specific to a device
- ▶ How to program it?

App Data Saving

- In BasicPage1.xaml

```
<TextBox Name="nameInput" Width="300" HorizontalAlignment="Left" TextChanged="nameInput_TextChanged"/>
```



- In BasicPage1.xaml.cs (without MVVM)

```
private void nameInput_TextChanged(object sender, TextChangedEventArgs e)
{
    var app = Windows.Storage.ApplicationData.Current;
    var roamingSettings = app.RoamingSettings;
    roamingSettings.Values["nameInput"] = nameInput.Text;
}
```

Session Data Saving

- Data
 - Relative to the active session of the user
 - Could be restored later if Windows suspends and stops the app
- The developer must register
 - The navigation state of the frame of the app so that
 - The app can be restored at the page where it was
 - The SuspensionManager can identify the page where the state must be restored
 - The state of the page itself

Session Data Saving

- ▶ How to program it?
- ▶ Use SuspensionManager
 - ▶ App.xaml.cs contains an administrator for the **App.Suspending** event called when Windows is about to suspend the App

```
public App()  
{  
    this.InitializeComponent();  
    this.Suspending += this.OnSuspending;  
}
```

Session Data Saving

- In the App.xaml.cs,
 - In the method OnSuspending,
 - The calling of the SaveAsync method
 - Register the state of navigation of the Frame
 - And allow the page to register its contents (by the method navigationHelper_SaveState)

```
private async void OnSuspending(object sender, SuspendingEventArgs e)
{
    var deferral = e.SuspendingOperation.GetDeferral();

    await SuspensionManager.SaveAsync();

    deferral.Complete();
}
```

Session Data Saving

- In BasicPage1.xaml.cs, add some code in the *navigationHelper_SaveState* method to record the state of the page

```
private void NavigationHelper_SaveState(object sender, SaveStateEventArgs e)
{
    e.PageState["messageOutputText"] = messageOutput.Text;
}
```

- Serializes and registers the *PageState* dictionary in an XML file
- Registered data in *PageState* only recorded for this session

App State and Session Data Restoring

➤ Step 3 : Restore the App state

- In the App.xaml.cs, call of *RestoreAsync* to restore the navigation state of the frame and to allow the page to restore its contents

```
protected async override void OnLaunched(LaunchActivatedEventArgs e) {  
    ...  
    if (rootFrame == null)  
    {  
        .....  
        if (e.PreviousExecutionState == ApplicationExecutionState.Terminated)  
        {  
            await SuspensionManager.RestoreAsync();  
        }  
        ...  
    }  
}
```

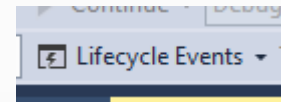
App State and Session Data Restoring

- ➔ In BasicPage1.xaml.cs,

```
private void NavigationHelper_LoadState(object sender, LoadStateEventArgs e)
{
    // Restore values stored in session state
    if (e.PageState != null && e.PageState.ContainsKey("messageOutputText"))
    {
        messageOutput.Text = e.PageState["messageOuputText"].ToString();
    }
    // Restore values stored in app data
    var app = Windows.Storage.ApplicationData.Current;
    var roamingSettings = app.RoamingSettings;
    if (roamingSettings.Values.ContainsKey("nameInput"))
    {
        nameInput.Text = roamingSettings.Values["nameInput"].ToString();
    }
}
```

Debugger

- Simulation of the suspension in Visual Studio
- Press F5 to execute the app in debugging mode
- Enter some data on the page
- Press Alt+Tab to return in Visual Studio
- Open the drop-down menu
 - Select Suspend and shutdown
 - The event Suspending occurs
 - The management state code is executed



Debugger

- Press F5 to re-execute the App
 - The app is restored in the previous state
- Modify the text field, click on the button
- Press on Alt+Tab to return in Visual Studio
- Close the App by selecting Debug > Stop the debugging
- Press F5 to re-execute the App
 - At the moment, the student name is restored, because it was registered as the user typed it
 - The message is not restored, because the Suspending event did not occur and thus the state of session was not registered