

Hands-on lab

Lab 5: Process Lifetime Management

September 2012



Contents

[Overview 3](#_Toc334626994)

[Exercise 1: Navigation state is saved for you 4](#_Toc334626995)

[Exercise 2: Tips 5](#_Toc334626996)

[Summary 6](#_Toc334626997)

Overview

* 1. Process Lifetime Management, or PLM, is one of the most important concepts for a developer building Windows Store apps to understand. Unlike traditional Windows applications, which continue to execute even when they are in the background, Windows Store apps execute only when they are in the foreground. (Note that snapped apps still count as foreground.) In general, an app that is not visible to the user is suspended by the operating system and is unable to execute until the OS restores it to the foreground.
  2. When an app is suspended, it remains in memory with all its threads suspended. As long as the process remains in memory, the application resumes executing where it left off when it comes back to the foreground. There is nothing you, the developer, have to do to make that happen. However, you are not *guaranteed* that the process will remain in memory. If, while an app is suspended, the Windows kernel determines that it needs the memory the suspended app is consuming, Windows can *terminate* the suspended app. When an app is terminated, all state that has not been saved is lost. Unless you take steps to preserve that state, the user will be in for a surprise when he or she swaps back to the application and finds that all the work he or she has done is gone.
  3. Obviously, we cannot allow this to happen. At the time the application is suspended, even Windows doesn’t know whether the app will eventually be terminated. You should assume that your app will be terminated, so you must write code to save the state of the app when it is suspended. Then you can restore the state upon reactivation if you detect that a termination occurred while the app was suspended.
  4. Sound tricky? It can be sometimes, but for most apps it is not and it is work you have to do to make a great app. The cost of not doing so is the potential for a user to lose everything he or she has done in *your a*pp simply by switching briefly to another application.

**Objectives**

* 1. This lab will introduce you to PLM and show you some of the code that Visual Studio writes for you in the default templates. Because Contoso Cookbook is a small app, we don’t have have to write new code to handle PLM; the code that Visual Studio injects is enough. You are likely to have to write more code for your own apps, though, so we’ll walk you through where the code is and what APIs you can use. Sit back and take a break from typing, but do pay attention.

**System requirements**

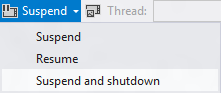
* 1. You must have the following items to complete this lab:
  + Microsoft Windows 8
  + Microsoft Visual Studio 2012

**Setup**

* 1. To prepare your computer for this lab, you must:
  2. Install Microsoft Windows 8.
  3. Install Microsoft Visual Studio 2012.

**Exercises**

* 1. This nands-on lab includes the following exercises:
  2. Navigation state is saved for you.
  3. Tips

1. Exercise 1: Navigation state is saved for you
   1. Because the only state that needs to be saved in Contoso Cookbook is the navigation state—the item or group the user was viewing and the navigation history—you don’t have to do anything to handle PLM. Visual Studio included a class named **SuspensionManager** in your app; it is located in SuspensionManager.cs in the project’s Common folder. Visual Studio also included a line of code in the **App** constructor in App.xaml.cs that registers a handler for **Suspending** events. The handler—**OnSuspending**—calls **SuspensionManager.SaveAsync** to save the app’s navigation state as shown here.
   2. private async void OnSuspending(object sender, SuspendingEventArgs e)
   3. {
   4. var deferral = e.SuspendingOperation.GetDeferral();
   5. await SuspensionManager.SaveAsync();
   6. deferral.Complete();
   7. }
   8. Visual Studio included an **if** clause in the **OnLaunched** method in App.xaml.cs that restores the app’s navigation state if the app was terminated by the operating system after it was suspended.
   9. if (args.PreviousExecutionState == ApplicationExecutionState.Terminated)
   10. {
   11. // Restore the saved session state only when appropriate
   12. await SuspensionManager.RestoreAsync();
   13. }
   14. The result of all this is that you get a lot for free. If Contoso Cookbook is suspended and terminated, it automatically goes back to the last page you were viewing when it is restarted. You can test this by starting the app from Visual Studio by pressing F5, selecting a recipe, and selecting **Suspend and shutdown** from the **Debug Location** toolbar as shown here.
   15. 

After shutting the application down in this way, press F5 to restart the app. This simulates what happens when the app is terminated and restarted by the operating system. Thanks to the code created by Visual Studio, the app goes back to the recipe that you were viewing when you shut the app down. Because the navigation history was also restored, you can even use the **Back** button to retrace your steps through the app.

1. Exercise 2: Tips
   1. We highly recommend that you become familiar with this **Suspend**, **Resume**, and **Suspend and shutdown** commands in Visual Studio. It is a great way for you to debug your app’s PLM state. If you try to debug PLM just by starting the app and task-switching so the app moves in and out of the foreground, you’ll get good results most of the time. For termination, however, you will not see the right behavior; you’ll see the app remain in memory longer than it should because the Visual Studio debugger does a bit of magic to stay attached to the app. For termination events, use the Visual Studio **Suspend and shutdown** command.
   2. Deciding what to save for PLM is very important too. In the next lab, we will talk about Settings (or user preferences) and you’ll notice that we will save those immediately when the user enters or changes them within the app. The state you save for PLM should only be a ”transient” state that helps you restore users to where they were before the app was suspended. Anything else that needs to be saved (such as settings or cache) should be saved on demand as early as possible. This practice reduces the amount of work you have to do during the PLM suspending event.

Summary

* 1. Handling **Suspend** and **Resume** events for PLM is a requirement for a great Windows Store app. In our Contoso Cookbook lab, we did not have to do a lot of work to do this because all we were saving was navigation history. In your apps, you will likely have to do a little more. This lab has introduced you to the code that VS generates and the WinJS infrastructure for PLM. There is still more to do on Contoso Cookbook, though. Our next lab will show you how to save user preferences using the Settings charm. Let’s keep going.