CHAPTER 5

Life-Cycle Events

In this, the final chapter in this book, I show you how to take control of the Windows 8 app life cycle by responding to key Windows events. I show you how to replace code that Visual Studio adds to projects, how to properly deal with your app being suspended and resumed, and how to implement contracts that tie your app into the wider user experience that Windows 8 offers. Along the way, I’ll demonstrate the use of the geolocation feature and show you how to set up and manage a recurring asynchronous task. Table 5-1 provides the summary for this chapter.

Table 5-1. Chapter Summary

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
<th>Listing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure that your app receives the suspending and resuming events.</td>
<td>Subscribe to events from the Windows.UI.WebUI.WebUIApplication object.</td>
<td>1</td>
</tr>
<tr>
<td>Create a recurring background task.</td>
<td>Use the WinJS.Promise object as a wrapper around other asynchronous activities.</td>
<td>2–4</td>
</tr>
<tr>
<td>Request more time before your app is suspended.</td>
<td>Call the suspendingOperation.getDeferral method on the event passed to your suspending handler function.</td>
<td>5</td>
</tr>
<tr>
<td>Implement a contract.</td>
<td>Declare the contract in the manifest and respond to the type information in the activation event.</td>
<td>6–8</td>
</tr>
</tbody>
</table>

Dealing with the App Life Cycle

In Chapter 1, I showed you the skeletal code that Visual Studio placed into the default.js file to give me a jump-start with my example project. This code handles the Windows 8 application life-cycle events, ensuring that I can respond appropriately to the signals that the operating system is sending me. There are three key stages in the life of a Windows 8 app.

The first stage, activation, occurs when Windows wants your app to perform some task. The most common task is when Windows wants to launch your app and display it to the user, but there are other tasks as well, and you’ll see one example when I show you how to implement a contract later in this chapter.
Users don’t close Windows 8 apps; they just move to another application and leave Windows to sort things out. This is why there are no close buttons or menu bars on a Windows 8 UI. A Windows 8 app that is no longer required is moved into the second stage and is **suspended**. While suspended, no execution of the app code takes place, and there is no interaction with the user.

If the user switches back to a suspended app, then the third stage occurs: the application is **resumed**. The app is displayed to the user once execution continues. Suspended applications are not always restored. If the device is low on memory, for example, Windows may simply **terminate** a suspended app.

**Correcting the Visual Studio Event Code**

The code that Visual Studio adds to a new project is sufficient to get a basic app up and running, but it doesn’t support the full range of life-cycle events. It deals with activation and suspension quite happily, but it prevents the application from being notified when it is being resumed. Fortunately, there are other places in the API where I can register to receive the life-cycle events, so my first task in this chapter is to update `default.js` so that I am properly notified when my app enters all three life-cycle stages. You can see the changes in Listing 5-1.

**Listing 5-1.** Registering for Life-Cycle Event Notifications

```javascript
(function () {
  "use strict";

  Windows.UI.WebUI.WebUIApplication.addEventListener("activated", performInitialSetup);
  Windows.UI.WebUI.WebUIApplication.addEventListener("resuming", performResume);
  Windows.UI.WebUI.WebUIApplication.addEventListener("suspending", performSuspend);

  function performInitialSetup(e) {
    WinJS.UI.processAll().then(function () {
      UI.List.displayListItems();
      UI.List.setupListEvents();
      UI.AppBar.setupButtons();
      UI.Flyouts.setupAddItemFlyout();

      ViewModel.State.bind("selectedItemIndex", function (newValue) {
        WinJS.Utilities.empty(itemDetailTarget)
        var url = newValue == -1 ? "/html/noSelection.html" : "/pages/itemDetail/itemDetail.html"
        WinJS.UI.Pages.render(url, itemDetailTarget);
      });

      WinJS.UI.Pages.render("/html/storeDetail.html", storeDetailTarget);
    });
  }

  function performResume() {
    ViewModel.State.unbind("selectedItemIndex");
  }

  function performSuspend() {
    ViewModel.State.unbind("selectedItemIndex");
  }

  performSuspend();
})(jQuery);
```