Adventures in UI Automation: Scraping the Driver Files dialog

devblogs.microsoft.com/oldnewthing/20160314-00

March 14, 2016

Raymond Chen

If you go to Device Manager, then view the Properties of a device node, and then go to the *Driver* tab and click *Driver Details*, you get a dialog box with a list of driver files, and when you click each file, you get information about it.

Unfortunately, this dialog does not have a *Copy* button to copy the information to the clipboard. This means that when somebody on the hardware team asks you for the driver information, you have to manually copy down all the drivers and versions.

And by “you” I mean “me”.

As the joke goes, a programmer is someone who will spend six months writing a computer program to save 45 minutes of work. (Obligatory XKCD.)

I’m a programmer. Let’s start.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Automation;
using System.Runtime.InteropServices;

static class AutomationElementHelpers
{
 public static AutomationElement
 FindChildById(this AutomationElement parent, int id)
 {
  return parent == null ? null :
   parent.FindFirst(
    TreeScope.Children,
    new PropertyCondition(AutomationElement.AutomationIdProperty,
                          id.ToString()));
 }

 public static IEnumerable<AutomationElement>
 EnumChildrenOfControlType(this AutomationElement parent, ControlType type)
 {
  return parent == null ? Enumerable.Empty()
                        : parent.FindAll(TreeScope.Children,
   new PropertyCondition(AutomationElement.ControlTypeProperty,
                         type)).Cast<AutomationElement>();
    }
}
```

The `FindChildById` method looks for a child item with a specific automation ID.

The `EnumChildrenOfControlType` returns all the children of an automation element with the specified control type.

The last little helper thingie I need is a p/invoke.

```
static class Win32
{
 [DllImport("user32.dll", EntryPoint = "FindWindow", SetLastError = true)]
 static extern public System.IntPtr
 FindWindowByName(System.IntPtr MustBeZero, string name);
}
```

Technically, I didn't need this helper function; I could have used `AutomationElement-Helpers.Find`, but going straight for the window by title is faster because avoids the full search of the automation tree.

Now let's snap some blocks together.

```
class Program
{
 [System.STAThread]
 public static void Main(string[] args)
 {
  // Find the Driver File Details dialog.
  var dialog = AutomationElement.FromHandle(
   Win32.FindWindowByName(IntPtr.Zero, "Driver File Details"));

  // Find the various pieces of the dialog.
  var list = dialog.FindChildById(228);
  var provider = dialog.FindChildById(229);
  var version = dialog.FindChildById(230);
  var copyright = dialog.FindChildById(231);
  var signer = dialog.FindChildById(232);

  // Enumerate and print the list items
  foreach (AutomationElement item in
          list.EnumChildrenOfControlType(ControlType.DataItem))
  {
   System.Console.WriteLine("Driver: {0}", item.Current.Name);

   var pattern = item.GetCurrentPattern(SelectionItemPattern.Pattern)
                 as SelectionItemPattern;
   pattern.Select();

   System.Console.WriteLine("Provider: {0}", provider.Current.Name);
   System.Console.WriteLine("Version: {0}", version.Current.Name);
   System.Console.WriteLine("Copyright: {0}", copyright.Current.Name);
   System.Console.WriteLine("Signer: {0}", signer.Current.Name);
   System.Console.WriteLine();
  }
 }
}
```

Okay, let's see what just happened.

First we find the *Driver File Details* dialog box with `FindWindow`. As I noted, we could have used `AutomationElementHelpers.Find`, but the `FindWindow` is much faster because it doesn't have to search the entire automation tree. (And I got tired of waiting for the tree search each time I wanted to test the program.)

Once we find the dialog, we locate the important pieces of the dialog via their automation ID:

- The control that has the list of driver files.
- The controls that report on the properties of the selected driver file.

Note that automation IDs are not contractual; they can change at any time. But then again, automation is not contractual either, because the user interface can change at any time. It's handy for little tools like this, but you have to be aware that the automation may need to be updated if the user interface changes.

We enumerate all the children of the list which say that they are a data item. This filter may seem unnecessary to the casual observer, but it’s important, because a device with a huge number of driver files—I’m looking at you, the display driver—will probably end up showing a scroll bar, and we don’t want to enumerate that.

For each item, we select it, and then read out the values for the provider, version, copyright, and signer.

And that’s it. A program to extract the contents of the *Driver Files* dialog.

Raymond Chen

Follow