



## Controlling Modal Dialogs

application only if the dialog closes itself after a while. I know how to do this too.

I just call the `IHTMLWindow2.showModalDialog` myself.

3. Let the dialog show and interact with its DOM. This is my problem. The `IHTMLWindow2.showModalDialog` doesn't give me access to the dialog's DOM (as far as I know). I thought about the following workarounds.

Any other suggestions would be most welcome.

A. Use `IHostDialogHelper.ShowHTMLDialog`. This may be good if the last parameter to the function (`IUnknown *punkHost`) gives me access to the DOM somehow. I was unable to find any documentation about this parameter.

B. Host MSHTML myself for the dialog's HTML. I am not sure this would work if the script in the dialog would access members of the `IHTMLDialog` interface that is implemented by the window object created by `IHTMLWindow2.showModalDialog`.

C. Create a new tab and new `CustomWebBrowser`, as I do for `window.open` call, and have it navigate to the dialog's URL.

About this solution I am not sure how to handle the modality considerations.

Other tabs in my application need to be disabled. My implementation of `showModalDialog` needs to return only when the created browser exits. Again, what would happen if (when) the script calls items in the `IHTMLDialog` interface, like `IHTMLDialog.returnValue`.

Any help would be greatly appreciated.

Finally, here are the details of my implementation.

In my derived

```
class CustomWebBrowser : System.Windows.Forms.WebBrowser
```

In the constructor I set the `ObjectForScripting` as below. This object has an implementation of a method named `showModalDialog`, after assigning the `ObjectForScripting` property this method is available to script through the `window.external.showModalDialog` reference. Now all that remains in order to hook this method is to replace the script's `window.showModalDialog`.

```
This is done in the implementation of DWebBrowserEvents2.DocumentComplete method by executing javascript code that replaces the window.showModalDialog with 'function(dialog, varArgIn, varOptions){ return window.external.showModalDialog(dialog, varArgIn, varOptions); }
```

See the main implementation details below.

```
public class CustomWebBrowser : System.Windows.Forms.WebBrowser
```

## Controlling Modal Dialogs

```
{
public CustomWebBrowser()
: base()
{
this.ObjectForScripting = new ScriptingCallableObject(this);
}
....
protected override void CreateSink()
{
base.CreateSink();
m_eventsImplementation =
new DWebBrowserEvents2Implementation(this);
eventsCookie =
new AxHost.ConnectionPointCookie(this.ActiveXInstance,
m_eventsImplementation,
typeof(DWebBrowserEvents2));
}
....

public class DWebBrowserEvents2Implementation : DWebBrowserEvents2
{
public DWebBrowserEvents2Implementation(CustomWebBrowser browser)
{
m_Browser = browser;
}
....
[DispId(259)]
public void DocumentComplete(object pDisp, ref object URL)
{
IHTMLDocument2 doc2 = m_Browser.IE.Document as IHTMLDocument2;
doc2.parentWindow.execScript("if
(typeof(gpfn_OrigShowModalDialog) == \"undefined\") {
gpfn_OrigShowModalDialog = window.showModalDialog; window.showModalDialog =
function(dialog, varArgIn, varOptions){ return
window.external.showModalDialog(dialog, varArgIn, varOptions); }; }",
"javascript");
}
....
}

....
[ComVisible(true)]
[ClassInterface(ClassInterfaceType.AutoDual)]
public class ScriptingCallableObject
{
public ScriptingCallableObject(CustomWebBrowser browser)
{
m_WebBrowser = browser;
}
}
```

## Controlling Modal Dialogs

```
public object showModalDialog(string dialog, object varArgIn,
string varOptions)
{
// This implementation preserves the default behavior.
// but I can implenet whatever I want here.
IHTMLDocument2 doc2 = m_WebBrowser.IE.Document as
IHTMLDocument2;
object options = varOptions;
return doc2.parentWindow.showModalDialog(dialog, ref
varArgIn, ref options);
}

private CustomWebBrowser m_WebBrowser;
}
}
```

---

Adar Wesley