How to customise the TWebBrowser user interface (part 2 of 6)

Overview of a solution

Overview of a solution

Let us assess what are trying to do. We will work towards creating an object that can customize the *TWebBrowser* control's appearance and take control of its context menu. In the interests of ease of use and reusability we will expose properties to enable users to easily configure these attributes of the browser control.

So how do we go about customizing the *TWebBrowser*? According to the Microsoft® Developer Network Library documentation (my emphasis):

> *"The mechanism for WebBrowser Control customization is designed to be automated when a **container** provides support for ActiveX controls. Whenever the WebBrowser Control is instantiated, it attempts to find **IDocHostUIHandler**, IDocHostUIHandler2 and IDocHostShowUI implementations from the **host**, if they are available. The WebBrowser Control does this by a QueryInterface call on the host's **IOleClientSite** interface.*
>
> *"This architecture works automatically for an application that implements an IOleClientSite interface and that passes an IOleClientSite pointer to the WebBrowser Control through the browser's IOleObject::SetClientSite method."*

This tells us the following:

- We need to create a "container" object to host our *TWebBrowser* control.
- This container must implement the *IOleClientSite* interface to enable the browser control to query it when looking for our *IDocHostUIHandler* interface.
- We need to notify the *TWebBrowser* that we are hosting it by passing a reference to our container's *IOleClientSite* interface to the browser control via its *IOleObject.SetClientSite* method.
- The container object must also implement *IDocHostUIHandler*. This implementation must provide the required customization of the browser control. (We will not concern ourselves with *IDocHostUIHandler2* or *IDocHostShowUI* here).

In addition, our container class could expose properties to be used by *"client code"* to control various aspects of the browser's customization.

A few design decisions

We have establised that we need to develop a container control that supports *IOleClientSite*, *IDocHostUIHandler* and, by implication, *IUnknown*. The object will also expose customization properties.

Client code will instantiate the container object and manipulate its properties in the usual way. The web browser control will also access the container, but will do so via its supported interfaces. This means we will be mixing two different methods of access – by interface and by object reference. Whenever this is done on an object that supports reference counted interfaces there is a danger of the object being prematurely freed when an interface reference goes out of scope. Consequently we will manage the object's life time ourselves, i.e. it will not be reference counted.

In the interests of reusability we will develop two classes:

- *TNulWBContainer* – a do-nothing container object that hosts the web browser control and implements all the required interfaces in way that

Reusing the code

has no visible effect on the web browser control. This provides a base class for classes that seek to customize *TWebBrowser* in different ways. The intention is that this base class can be reused.

See *article #22*, "How to call Delphi code from scripts running in a TWebBrowser" for an example of how we reuse *TNulWBContainer*.

- *TWBContainer* – a descendant of *TNulWBContainer* that adds all the customization we noted in the *introduction*.

In the following two sections we will develop the code for these classes. We begin in the *next section* with a discussion of *TNulWBContainer*.
