How to customise the TWebBrowser user interface (part 1 of 6)

Contents and introduction

Contents

Introduction

Among the most popular questions posed in the *TWebBrowser* newsgroups and discussed on many Delphi tips sites are:

1. How do you get a *TWebBrowser* control to display the popup menu assigned to its *PopupMenu* property instead of the standard IE popup menu?
2. How do you stop *TWebBrowser* displaying 3D borders when a document is loaded.
3. How do you prevent *TWebBrowser* from displaying scrollbars?

There are a three other, related, questions that we will answer in this article. They arose when I was developing the *CodeSnip Database Viewer*:

1. How can you make the browser control take on the look and feel of the hosting form, given that each user may have a different display configuration that is not known at design time? I needed to do this to add some HTML content to a couple of dialog boxes and make the HTML appear to be part of the dialog box.
2. Users can normally select text in a browser control – something you don't want in a dialog box. So how do you stop that?
3. How do you ensure that the browser control uses themes when rendering widgets such as buttons when, and only when, your application is using themes?

The answer to the first question is to create an object that implements the *IDocHostUIHandler* interface and use the object to control the popup menu.

A common solution to the second and third questions is to set one of the DOM object properties to some suitable value. For example, if *WB* has type *TWebBrowser*, we must wait until a document has loaded and then do:

```
// Switch off scrollbars
WB.OleObject.document.body.style.overflowX := 'hidden';
WB.OleObject.document.body.style.overflowY := 'hidden';

// Switch off borders
WB.OleObject.document.body.style.borderstyle := 'none';
```

Listing 1

Now, what is less well known is that we can also set the border style and the scroll bar visibility by implementing *IDocHostUIHandler*.

IDocHostUIHandler also lets us provide a default Cascading Style Sheet (CSS) to the browser object. This provides an answer to the fourth question: we can dynamically create a style sheet that knows about a form's colour and fonts and tell the browser to use it.

And as for the last two questions? You guessed it: *IDocHostUIHandler* can help here too!

Unsurprisingly then, this article is all about how to create an object that implements *IDocHostUIHandler* and use it to customize the browser control. Along the way we will also have to find out how to get *TWebBrowser* to know about, and use, the object.

We begin the *next section* with an overview of our solution.
