

FormSet

A Delphi Custom Control

Overview

FormSet is a custom control based on the Delphi **TabSet**. You can install it on the tool palette through Options / Install Components on the Delphi action bar menu. I haven't learned how to write Windows help yet, so this file is all the help you get.

The class **TFormSet** is derived from **TTabSet**. All the properties, methods and events documented for **TTabSet** also work with **TFormSet**. With a **TabSet** you probably map each tab to a page in a **NoteBook**. With a **FormSet** you map each tab to a form. The main idea is to create smaller, more independent units and source files. Forms can be farmed out to different programmers on a team and reused much more easily than notebook pages.

This document calls the form that has the **FormSet** the *Parent* or *Book* form, and the forms displayed with the tabs *Child* or *Page* forms. When you click on a tab, the **FormSet** displays the *Child* form in the client area of the *Parent*. When you change tabs, the **FormSet** can release the old form for best memory use, or leave it in memory for best performance.

Design Time

In the designer, you put a **FormSet** on the *Parent* form just like a **TabSet**. You may specify the tab captions in the *Tabs* array at design time or add them at run time via `FormSet.Tabs.Add('Caption')`.

Child or *Page* forms are as normal as I could make them. At design time, *Visible* must be *False*, and *BorderStyle* must not be *bsNone*. You can leave page forms in the AutoCreate list under Options / Project, or you can remove them from the list and create and destroy them on the fly for memory management.

This document describes events fired on the *Page* forms. These are normal Delphi events, the same as for any *TForm*. You can put code in them or not.

Run Time

Some time before the *Parent* form becomes visible, set **FormSet.TabIndex** to a positive value. This will trigger a **Change** event within **FormSet**, which is required to get the first tab and form displayed. If you do not modify **TabIndex** the tab captions will display but none will be "selected", and no *Child* form will be displayed.

There is one **FormSet** event which you must handle. It is called *OnTabLoad*. The **FormSet**

triggers this event any time a tab is clicked and the **FormSet** does not know the *Child* TForm object.

The first parameter for *OnTabLoad* is the usual Delphi *Sender*. The second is the *TabIndex* of the tab which is being loaded. The third parameter is an object called *TFormTab*, defined as follows:

```
FormTab = class
  Form      : TForm;
  OnOpen    : TFOpen;
  OnClose   : TFTClose;
end;
```

You must set the **Form** property to a TForm object or **Nil**. You can create the form on the fly or use an existing form object. In the following example, the forms for pages 1 and 2 are created on demand, while the form for page 3 is in the AutoCreate list. This assumes the forms for pages 1 and 2 are always released when closed; see the OnClose options below.

```
Procedure Form1.FormSet1OnTabLoad(Sender : TObject;
                                   TabIndex: Integer;
                                   TFormTab: TFormTab);
begin
  select case TabIndex
    case 0 : FormTab.Form := TPage1.Create(Self);
    case 1 : FormTab.Form := TPage2.Create(Self);
    case 2 : FormTab.Form := Page3;
    else   FormTab.Form := Nil;
  end;
end;
```

or if you prefer working with captions:

```
Caption := FormSet1.Tabs[TabIndex];
FormTab.Form := Nil;
if Caption = 'Home' then
  FormTab.Form := TPage1.Create(Self);
if Caption = 'Auto' then
  FormTab.Form := TPage2.Create(Self);
if Caption = 'Life' then
  FormTab.Form := Page3;
```

You can set the other properties of the **FormTab** object in **OnTabLoad** as well.

OnOpen can have one of the following values:

ftoSizePageToBook - Size the *Child* to match the *Parent*. This makes all *Child* forms the same size for a smooth appearance. **FormSet** saves the original size of the *Parent* as it opens the first form on a tab. If the user resizes the parent form and then switches to a tab with this attribute, **FormSet** restores the parent to its original size. Let me know if this makes sense, or if you'd rather size the child to the current size of the parent.

ftoSizeBookToPage - Size the *Parent* to match the *Child*. This could be used to accomodate one large *Child* without enlarging the *Parent* all the time.

OnClose can have one of the following values:

ftcAlwaysRelease - Release the form when changing to another tab

ftcNeverRelease - Do not release the form.

ftcMaybeRelease - Release the form later if memory runs low. Not implemented.

If you feel the urge to access **FormTab** objects directly, you can do so in any part of your code:

```
var
  FormTab : TFormTab;
begin
  FormTab := FormSet1.Tabs.Objects[TabIndex] as TFormTab;
  FormTab.Property := Value;
```

Events

With a Delphi **TabSet**, events are triggered in this order:

TabSet.OnChange
TabSet.OnClick

If the user clicks on a tab which is already active, neither event is fired. If the code in **OnChange** sets the *AllowChange* parameter to *False*, the change does not happen and *onClick* is not fired.

With a **FormSet**, events are triggered as shown below. This sequence may seem scrambled until you see the three major steps. First, **FormSet** checks to make sure the change is ok with all parties involved:

OldPage.OnCloseQuery - The old form can abort the change by setting *CanClose* to *False*; none of the other events in the list fire. It might do this if it has some unfinished business on screen.

FormSet.OnTabLoad - Called only if the form for the tab is not already known. You might choose to create a new form here. If **FormTab.Form** is **Nil** after the call to **OnTabLoad**, **FormSet** aborts the change and none of the other events fire.

NewPage.OnCreate - Triggered by your **Create** call from **OnTabLoad**, which in turn is called only if the form is not already known.

FormSet.OnChange - As with a normal **TabSet**, the application can abort the change by setting *AllowChange* to *False*. None of the other events fire.

Second, **FormSet** closes one form and opens another.

OldPage.OnHide - Always happens. Gives the old page a chance to save data.

OldPage.OnDestroy - Only if the old page has *OnClose* set to **fscAlwaysRelease**.

NewPage.OnShow - Always happens. Gives the new page a chance to load data.

Third, like the original **TabSet** control, **FormSet** fires the **Click** event after all the interesting stuff has happened. The new page is already visible by this time.

FormSet.OnClick

FSETDemo

FSetDemo is a program which demonstrates **FormSet**. As of the 6/2/96 version, some interesting events are written to a logger window instead of zillions of message boxes. I hope this doesn't confuse things too much. You might find it interesting to alter the logger window to have the "ShowMessage" box checked by default. This lets you look at each message and think about what's going on with **FormSet** before going on to the next.

Credits / Author

The idea of attaching forms to tabs is not original. I've seen it done nicely in other languages. The techniques for making it work in Delphi were worked out and posted on CompuServe by Neal Rubenking and Kevin Tupper. I tried to do this before I saw Neal's code, and got pretty close, but would have given up without the examples these guys put on CIS.

I put "forms on tabs" into a custom control. It was a bit of a bear. The *Component Writer's Guide* is a little thin on examples and **TabSet** is not included in the VCL source. I understand Borland obtained **TabSet** from a third party. It doesn't seem to follow the conventions for making subclassing easy, and I regret to say, I probably didn't improve things there.

FormSet is protected by copyright and is not public domain. It may be distributed freely. Please keep the following files together and unmodified:

FORMSET.PAS
FORMSET.DCU
FORMSET.WRI
FSETDEMO.ZIP

FormSet is a work in progress. This rough cut is made available to get some feedback. If you want to talk about how it was done, what works and what doesn't, please contact the author directly:

Jim Standley
493 Carnation Drive
Clarks Summit, PA 18411
Phone: (717) 586-1293 19:00 - 22:00 Eastern Time
E-Mail: 71631,305@CompuServe.com

6/10/96