EZForm

Delphi Class Library

Version 1.00

CLASSIC SOFTWARE LICENCE AGREEMENT

COPYRIGHT

Contents

Introduction 4
    What is EZForm? 4
    How does EZForm do it? 4
Installation 6
    Installing EZForm 6
    Installing the EZKeys VCL component 6
Conventions 7
Overview 7
    TcsEZForm 7
    TcsEZKeys 7
    TcsEZFormOptions 7
Using EZForm 8
    Creating new EZForms 8
    Adding EZForm behaviour to an existing form 8
Specifying the navigation keys for a particular form 9
    EZKeys VCL component 9
    NavigationKeys Property 9
Setting EZForm Options 10
    The EZFormOptions object 10
    csEZFormOptions Function 10
    Enabled Property 10
    DefaultNavigationKeys Property 11
Overriding EZForm behaviour for a specific type of control . . . 12
The Default Button 12
Contacting Classic Software 13

Introduction

What is EZForm?

EZForm is a set of Delphi classes which allow you to create forms which can be navigated using the Enter, Up/Down Arrow (* see note) and Ctrl+Tab keys in addition to the standard Tab key navigation. Because neither the Enter or Up/Down Arrow keys can be consistently used to move between every type of control the Ctrl+Tab keys are recognised by EZForms for this purpose.

* Note:

EZForms do not attempt to correct the anomalous behaviour already exhibited by Windows with regards to arrow key usage, e.g. being able to arrow from a button control (CheckBox, PushButton) into a control which has no tab-stop or when moving from a CheckBox or PushButton control into a RadioButton group. Only those keys not already used by the control are captured by EZForm, e.g. the arrow keys for Memo, ListBox, ComboBox, Button and Grid controls are not captured, the Enter key for PushButton, DBGrid and Memo (when WantReturns = True) controls are not captured etc.

How does EZForm do it?

The EZForm class automatically takes care of setting the appropriate Form properties/events (KeyPreview, OnKeyPress, OnKeyDown) and Component properties (Default) and calling specialised event handlers to achieve this behaviour. All you need to do to start using the EZForm capabilities is to specify that your form is an EZForm descendant, this can be done for a new or existing form. Default values are used for all properties to simplify use of the class.

With EZForm you do not need a new descendant control class for each different type of control that should respond to the enhanced navigation keys. EZForm has been designed to work (within the limitations outlined in the previous note) with all the existing standard Delphi controls and any descendants created from the standard controls.

In addition, EZForm has been designed so that it's capabilities can be easily enabled/disabled at run-time to allow your application to switch between standard and enhanced navigation keys, e.g. in response to a user changing a system option, without requiring two sets of forms/controls. Global enabling/disabling of EZForm enhanced navigation will be reflected immediately in all open EZForms, i.e. any previously defined Default buttons will be disabled/re-enabled and the form's behaviour will change immediately. Also, realising that you may not want all the enhanced
navigation keys to be active on a particular form, you can specify at design-time (using the EZKeys component) or in source code the navigation keys to allow.

All forms you create which are based on the EZForm class will automatically have enhanced navigation abilities, you don't need to make changes to each new form you create. You can add a blank form based on the EZForm class to your Forms Gallery which you can then use to easily create new EZForms.

No event handlers are reserved by EZForms; you can still assign your own methods to the form's OnKeyDown and OnKeyPress events. An EZForm automatically takes care of calling your event handlers before it's own. If necessary, your event handlers can override EZForm's event handlers.

Installation

Installing EZForm

To install EZForm you should make a new directory and copy all the files on the diskette into that directory. For example if you keep all your 3rd party libraries in subdirectories below the **c:\delphi\libs** directory then:

```
c:
md\delphi\libs\ezform
copy  a:*.*   c:\delphi\libs\ezform
```

Note: If you have received EZForm in zipped format (via e-mail) then use PKUNZIP to unzip the files into the appropriate directory.

If you keep your 3rd party library files elsewhere then substitute the appropriate drive and/or directory path.

It is recommended that with EZForm, and indeed with any other 3rd party libraries you may use, that you never change the supplied source code but instead create descendant classes if you need to alter the standard behaviour. In doing this you make it easier to apply any updates which may be supplied in the future. If you change the supplied source code you may be preventing yourself from using any future updates.

Installing the EZKeys VCL component

The EZForm library contains only one visual VCL component, the EZKeys VCL component. Note that it's use with EZForm is optional. Use the Options | Install Components command to install the CSEZForm unit, this will result in the EZKeys component being added to the Samples page. You can then use the Options | Environment | Palette command to move the EZKeys component to a different palette page if desired.

Conventions

By convention, all EZForm class types and public functions are given a prefix to prevent conflicts with other libraries or with your own source code. Types are given the prefix "Tcs", e.g. TcsEZForm, and functions are given the prefix "cs", e.g. csEZFormOptions.

Overview

The EZForm library consists of the following classes:

TcsEZForm

This class allows you to create forms which exhibit enhanced navigation behaviour using the Enter, Up/Down Arrows or Ctrl+Tab keys. The NavigationKeys property allows the set of recognised enhanced navigation keys to be changed.

TcsEZKeys

This class defines the EZKeys VCL component which allows the set of recognised enhanced navigation keys to be changed at design-time using the Object Inspector.

TcsEZFormOptions

An object of this class is created automatically and is used to store the EZForm Options that apply to all TcsEZForm objects. It has two properties; Enabled and DefaultNavigationKeys. You do not need to create any objects of this class. It is referenced using the function csEZFormOptions.

Using EZForm

Creating new EZForms

To allow easy creation of new EZForms it is recommended that you create a blank EZForm and add it to the Form Templates page of the Gallery. This can be done by following these steps:

1. Create a new blank form.
2. Add CSEZForm to the **uses** section.
3. Change the form's class from TForm to TcsEZForm.
4. Choose **Save As Template** from the form SpeedMenu.
5. Specify the appropriate information (File Name, Title, Description), for example "EZFORM", "EZForm" and "A blank EZForm".
6. Choose OK.

Now, assuming you have "Gallery: Use on New Form" checked on Environment Options | Preferences, to create a new EZForm all you need to do is to select the blank EZForm from the Gallery when you choose File | New Unit.

Adding EZForm behaviour to an existing form

Assuming that your existing form's class is TForm, all you need to do to add EZForm capabilities to the form is to add CSEZForm to the unit's **uses** section and change the form's class from TForm to TcsEZForm.

Alternatively, if your existing forms are based on a descendant of TForm you will need to change your base form class so it is a descendant of TcsEZForm instead of TForm.

For example if you have:

```
TMyBaseForm = class(TForm)         |
...                                |      Base Form Class
```

and also:

```
TForm1 = class(TMyBaseForm)
...
TForm2 = class(TMyBaseForm)
...
```

Then you will need to change your base class definition to:

```
TMyBaseForm = class(TcsEZForm)     |
...                                |      Base Form Class
```

All your forms will then have EZForm capabilities.

Specifying the navigation keys for a particular form

If necessary, you can specify the set of enhanced navigation keys to be recognised by a particular instance of an EZForm by using an EZKeys component on the form or by using source code to change the NavigationKeys property of the form.

Note: Unless the enhanced navigation keys to be recognised by the form are different to the default set (see DefaultNavigationKeys) you do not need to use either the EZKeys component or the NavigationKeys property.

EZKeys VCL component

The EZKeys VCL component can be used to specify at design-time which enhanced navigation keys are to be supported by the form. It is provided because there is a limitation in the current version of Delphi which prevents new properties of TForm descendants (as is TcsEZForm) from being visible/editable at design-time using the Object Inspector. Note that the EZKeys component is only of use on an TcsEZForm, it will have no effect on a normal TForm.

NavigationKeys Property

The NavigationKeys property of TcsEZForm can be used to specify the enhanced navigation keys for a particular EZForm instance. You would normally do this in the form's OnCreate event method by setting the NavigationKeys property. For example:

```
procedure TForm1.FormCreate(Sender: TObject)
begin
  NavigationKeys := [nkEnter];  { Enhanced navigation for Enter key only }
end;
```

Setting EZForm Options

The EZFormOptions object

There is one EZFormOptions object that applies to all current and future EZForm object instances. This object is created automatically by the CSEZForm unit and cannot be directly referenced outside this unit. It should be accessed using the csEZFromOptions function. The EZFormOptions object has the following properties:

Enabled
DefaultNavigationKeys

csEZFormOptions Function

This function allows the EZFormOptions object to be accessed outside the CSEZForm unit, it returns a reference to the existing EZFormOptions object. To use the function you should include CSEZForm in the **uses** clause of the unit in which it is being called.

Enabled Property

The boolean Enabled property can be used to change the enabled status of all current and future EZForms. By default this property is True. For example to immediately disable EZForm capabilities in all existing EZForms you would execute the following statement:

```
csEZFormOptions.Enabled := False;
```

Any EZForms created after executing this statement will also be "disabled", i.e. the enhanced navigation capabilties will be disabled but the forms themselves will still work but with standard form navigation.

DefaultNavigationKeys Property

The DefaultNavigationKeys property allows you to specify the default set of enhanced navigation keys (selected from Enter, Up/Down Arrows and Ctrl+Tab) to use if a form does not contain an EZKeys component and has not had specific enhanced navigation keys assigned in it's OnCreate event method. The property is a set type and requires (one or more) values from the set: [nkEnter, nkUpDnArrows, nkCtrlTab]. Specifying nkCtrlTab will allow both Ctrl+Tab (forward movement) and Ctrl+Shift+Tab (backward movement). By default, the value of this property is [nkEnter, nkUpDnArrows, nkCtrlTab]. You can use this property to limit the set of recognised enhanced navigation keys. For example if you want to only allow enhanced navigation for the Enter key for all your forms, you would execute the following statement when your application starts (before any EZForms are created):

```
csEZFormOptions.DefaultNavigationKeys := [nkEnter];
```

Changing this property will only be reflected in forms subsequently created, it has no effect on existing EZForms.

Note: To specify different enhanced navigation keys for a specific form instance you should use the NavigationKeys property of the TcsEZForm class, or the EZKeys component, rather than changing the DefaultNavigationKeys property.

Overriding EZForm behaviour for a specific type of control

If you have a particular type (class) of control for which you want to disable EZForm behaviour, this can easily be done. For example if you have a specialised Date Edit control, of class TMyDateEdit say, that uses the arrow keys to increment/decrement the date text, you can disable EZForm behaviour for this type of control by writing a new CanIntercept function in the form's class and having it override the function in TcsEZForm. Where you define the overridden function --- in your base class or in a particular form class --- will determine the extent of the overridden behaviour.

If we assume you have a base form class as discussed earlier then you could define:

```
type
      TMyBaseForm = class(TcsEZForm)
      ...
      public
         function CanIntercept(KeyType: TNavigationKey; Ctrl: TControl): Boolean;
                           override;
      end;


function TMyBaseForm.CanIntercept(KeyType: TNavigationKey; Ctrl: TControl):
Boolean;
begin
 if (Ctrl is TMyDateEdit) and (KeyType = nkUpDnArrows) then
   Result := False  { don't intercept arrows }
 else
   Result := inherited CanIntercept(KeyType, Ctrl);
end;
```

The Default Button

If your application will allow users to switch between standard (TForm) and enhanced (TcsEZForm) form navigation you can still specify a Default button when designing your forms. EZForms will take care of disabling the Default button status -- which is necessary to allow EZForm to intercept the Enter key -- when the form is created.

Contacting Classic Software

Classic Software can be contacted via CompuServe (preferred), telephone, fax and post.

CompuServe:	100033,1230
Internet:	100033.1230@compuserve.com
Telephone/Fax:	+61 9 271 5407
Mail:	Classic Software Unit 2/19A Wood Street INGLEWOOD WA 6052 AUSTRALIA