

## Re: Create Forms via Parameter

**Source:**

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.language.objectpascal/2003-12/1105.html>

---

**From:** Peter Below (TeamB) (100113.1101\_at\_compuXXserve.com)

**Date:** 12/12/03

Date: Fri, 12 Dec 2003 12:19:06 +0100

In article <3fd977b9@newsgroups.borland.com>, Arda Han wrote:

> *Anybody help me please! Peter Below where are you?*

Actually i'm at home suffering from some flu-like symptoms <g>.

> *My table structure :*

> *MenuName,FormName,Note,MenuParent,MenuID*

>

> *I want create dynamically menu and call assigned form from Table.*

>

> *I am create menu but I can't create Form. Ian'c code is not running? Help*

> *me please...*

If you store the forms name (the content of the Name property) you need to prefix a "T" to it to get the classname. You register form classes, so you need the classname to retrieve a class registered via RegisterClasses.

There are likely hundreds of examples of this stuff in the newsgroup archives, one of them is this:

<quote

source="<http://codecentral.borland.com/codecentral/ccweb.exe/listing?id=19374>">

Another take on creating forms "by name" or "by classname"

> *A fellow developer and I have a database in which is imbeded names of*

> *various forms.*

>

> *When we access the DB we can of course retrieve the name as as cmd\_text.*

>

> *Now we need to execute or run the value of cmd\_test.*

>

> *The best, but proolly not feasable in D7, would be able to instansiate a*

> *TAction for cmd\_text then run it from an action list.*

I don't see how you get the TAction into the picture, it will certainly not create a form for you automatically.

The problem to solve is how to get from a form \*name\* (or classname) held in a string to a form instance you can show. How to go about that depends a bit on how you create your forms.

a) the forms are all autocreated on application startup.

This is a very bad idea in large applications, since it makes your app consume a lot of resources which may never be needed, but it is the default setup you get when you don't tell the IDE not to autocreate forms (with the exception of the main form) for you.

In this scenario the form instances already exist, you just need to find them. Since all autocreated forms are owned by the Application object the best way to do this is:

```
Function ShowFormByName( const formname: String;
                        showit: Boolean = true ): TForm;
Var
  comp: TComponent;
Begin
  comp := Application.FindComponent( formname );
  If Assigned( comp ) and (comp Is TForm) Then Begin
    Result := TForm( comp );
    If showit Then
      Result.Show;
  End
  Else Begin
    Result := nil;
    Assert( false, 'Form '+formname+' not found' );
  End;
End;
```

If what you have is a class name, not a form name, i would iterate over the Screen objects Forms property:

```
Function ShowFormByClassname( const classname: String;
                              showit: Boolean = true ): TForm;
Var
  i: Integer;
Begin
  Result := nil;
  For i:= 0 to Screen.Formcount-1 Do
    If Screen.Forms[i].ClassNameIs( classname ) Then Begin
      Result := TForm( Screen.Forms[i] );
      If showit Then
        Result.Show;
      Break;
    End;
  Assert( Assigned( result ), 'Form '+classname+' not found' );
End;
```

Note that you do not use the form variable the IDE creates for you here. There is no way to find the address of that variable given its name, but you really don't need to do that. You need the \*content\* of the variable, the form reference, and that can be had the way shown above.

b) You create forms on demand and want to create a new instance of the form, given its classname.

If your database tables contain the form name (not the class name) you need to adhere to a naming convention that allows you to derive the class name from the form name. The IDE uses one already: just prefix a 'T' on the name and you have the classname.

To create an instance of a form we first need to identify the class and get a class reference for it, so we can call the Create constructor on that reference. For this we need a class registry that "knows" about the available form classes and which we can query for the class using its name. The VCL contains a suitable class registry, but the form classes are not registered automatically in it. This you need to do yourself. You do it by adding an Initialization section to every unit that implements a form you want to be able to create this way:

#### Initialization

```
RegisterClass( TfrmCustomerDetail );  
End.
```

If you have not worked with Initialization sections before: they go at the very end of a unit.

If the classes are all registered properly we can use functions similar to the ones above:

```
Function CreateFormByClassname( const classname: String;  
                               showit: Boolean = true ): TForm;  
Var  
  classref: TPersistentClass;  
Begin  
  classref := GetClass( classname );  
  If Assigned( classref ) and classref.InheritsFrom( TForm ) Then Begin  
    Result := TFormClass( classref ).Create( Application );  
    If showit Then  
      Result.Show;  
  End  
  Else Begin  
    Result := nil;  
    Assert( false, 'Form '+classname+' not found' );  
  End;  
End;  
  
Function CreateFormByName( const formname: String;  
                           showit: Boolean = true ): TForm;
```

```
Begin
  Result := CreateFormByClassname( 'T'+formname, showit );
End;
```

The caller takes ownership of the form reference returned by the functions. If it passes false for showit and then shows the form modally it should free it when the showmodal call returns:

```
with CreateFormbyClassname( classname, false ) do
try
  If showmodal = mrOK then
    ... do something
finally
  free
end;
```

You can of course also assign the returned reference to a local variable of type TForm to work with it.

If the form is shown modeless (pass true for showit or omit the parameter) it should have a handler for the OnClose event that sets Action := caFree, this way the form will self-destruct when the user closes it.

Again you do not use the IDE-created form variable. In fact you should delete this variable from the units source code if you create forms dynamically this way, it is no use and can lead to errors if you refer to it in your code, since it will never contain the form reference you need.

</quote>

```
--
Peter Below (TeamB)
Use the newsgroup archives :
http://www.mers.com/searchsite.html
http://www.tamaracka.com/search.htm
http://groups.google.com
http://www.prolix.be
```