

Re: Create objects using dynamic class names

Source: <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2006-08/msg00398.html>

- *From:* "Maarten Wiltink" <maarten@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 11 Aug 2006 14:36:48 +0200
-

"Boefje" <boefje@xxxxxxxxxxx> wrote in message
news:1155294676.167575.165660@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
[...]

I am trying to make a Wizard form. The idea is to build it in such a way it will be easy to use it in my other projects. It must be as general as possible, but without overdoing it.

This is useful background.

Have you thought about cases where there may be different paths through the scenario? I'm not saying you have to solve that completely right now, but it might be useful to brainstorm a little beforehand about required changes if your storyboard becomes a tree, or an acyclic graph.

One particularly useful trick that I'd never thought of myself until my colleague used it, is to keep a history stack in the wizard itself. That way, the Back button becomes trivial.

[...]

Now I come back to your proposed solution. You say: "A more elegant solution to your current problem could be to derive your TFrameStapX classes from a common base class and have an array of class references to the individual classes."

(You do realise that this is, on Usenet, a non-standard way of quoting?)

Would that be an open array? And can I create a class name (TFrameStapX) on the fly? Do I understand it right that I can call the constructor of the base class, which will create an instance of a sub class, based on the constructor's parameters?

Re: Create objects using dynamic class names

One at a time.

Not necessarily an open array, although that neatly coalesces the steps of telling the wizard class how many tabs you want, and which. But often, it's a constant array written out ahead of time, and you only specify the array index to the receiving code. That need not be an integer; often an enumerated value works very well.

A string value actually often works less well. Pascal is a strongly typed language; make that work `_for_` you! Strings are not the only datatype and not the best way to identify things that you know at compile time.

The constructor of the base class can only construct an instance of the base class. What you describe is known as a class factory – note that you can have a class method named `Create` in the base class, which is declared to return an instance of the base class and in truth always returns an instance of a derived class (it needs to decide which one through its parameters). It may look like a constructor, but it isn't.

But you can make a constructor of the base class virtual, and call it through a class reference (taken from that array). Every derived class can override it (if it needs to) and the 'most recent' version will be called, depending on the class reference.

Let's make a few assumptions now, since details depend on them. Let's assume that you will reuse the wizard class and the page base class in several programs, but that you do so by putting them in a package or including them in the project each time.

Then the set of available pages, and their behaviour, will be known in advance in each project. So you can have declarations and code like the following:

```
type
TPage = class;
TPageClass = class of TPage;

TWizard = class(...)
private
FPages: TObjectList;

function GetPage(const Index: Integer): TPage;
function GetPageCount: Integer;
protected
property Page[const Index: Integer]: TPage read GetPage;
property PageCount: Integer read GetPageCount;
public
procedure Init(const PageClasses: array of TPageClass; ...);
end;
```

Re: Create objects using dynamic class names

```
TPage = class(...)
public
constructor Create(...); virtual; abstract;
end;

....

function TWizard.GetPage(const Index: Integer): TPage;
begin
Result:=FPages.Items[Index] as TPage;
end;

function TWizard.GetPageCount: Integer;
begin
Result:=FPages.Count;
end;

procedure TWizard.Init(const PageClasses: array of TPageClass; ...);
var Index: Integer;
begin
FPages.Clear();

for Index:=Low(PageClasses) to High(PageClasses)
do FPages.Add(PageClasses[Index].Create(...));
end;

....

type
TFirstPage = class(TPage)
public
constructor Create(...); override;
end;

TLastPage = class(TPage)
public
constructor Create(...); override;
end;

....

Wizard.Init([TFirstPage, TLastPage], ...);
```

In the end, I've never used classnames in strings and there isn't even a central list of which classes are available and through which identifiers. The first is not a surprise; it's rarely needed. Who needs class names in a language where you can have class references? The second is a consequence of all page classes being fully known at compile time and it being easier to construct a temporary list on the fly.

Re: Create objects using dynamic class names

Groetjes,
Maarten Wiltink

.