

## Re: MD7 Q's

**Source:** <http://coding.derkeiler.com/Archive/Delphi/comp.lang.pascal.delphi.misc/2003-10/0395.html>

---

**From:** J French ([erewhon\\_at\\_nowhere.com](mailto:erewhon_at_nowhere.com))

**Date:** 10/13/03

Date: Mon, 13 Oct 2003 11:28:53 +0000 (UTC)

On Mon, 13 Oct 2003 11:26:22 +0200, "MW" <[mwdejager@hotmail.com](mailto:mwdejager@hotmail.com)> wrote:

>Thanks Rob

>

>Much of your explanations here, have answered and confirmed my findings, on  
>many a question I've had myself. You answer a few more questions later on  
>the thread regarding the freeing of objects, and I have still some more  
>questions.

>

>Say you define a TButton and instantiate it as follows

>

>procedure TForm1.FormMouseDown [..]

>var

> Btn: TButton;

>begin

> Btn := TButton.Create (Self);

>end;

>

>Am I corrent in assuming that this instance of TButton, Btn, only gets freed  
>once I close Form1? (unless if I call the free/destroy method myself)

Yes – its Owner will destroy it

(failing that it should die with the App – but that is a digression)

Owners kill things – App are meta-owners

>

>Also, which properties of Btn are set by TButton.Create, if any, or do I

>need to populate all of its properties?

A Parent helps a lot ...

>

>Now say, I went and instantiated an object of type TStringList.

>

>procedure StoreNames;

>var

> Names : TStringList;

>begin

> Names := TStringList.Create;

> With Names Do

> Begin

```
> Clear; { is this necessary }
> Add('Rob');
> Add('name');
> Add('MW');
> End;
> ...
> ...
> // ClearList(Names);
>
> Names.Free;
> end
>
> Does the above code (Names.Free) take care of cleaning up properly? I've
> read some code where the programmer did cleaning up that freed each of the
> strings that were added individually. (commented out ClearList)
```

They are wrong

Kill a TStringList and the Strings go

```
>
> procedure ClearList(List: TStringList);
> var
> i : integer;
> begin
> for i := 0 to List.Count - 1 do
> if List.Objects[i] <> nil then
> List.Objects[i].Free;
> List.Clear;
> end;
```

```
>
> Is the above procedure really necessary?
```

Objects are something different

– they are actually 4 bytes that point to pretty much anything

Whether or not to Free them is the programmers choice

– IMO an object in a TList with no other 'associations' is dangerous

```
>
> Also, what happens to Names, if I do not call Free. Does Delphi not clean
> up for a local variable of type TStringList, like it would have for a
> variable of type String, when the procedure closes?
```

Nope – it lingers and festers

```
>
> Thank you in advance for any answers
```

It will be interesting seeing alternative views.

```
>
> Kind regards
```

```
>
> Groete
> MW
```

```
>
>
> "Rob Kennedy" <. > wrote in message news:vogn97e929l5f6@corp.supernews.com...
>> name wrote:
```

```
>>> --- Self/Parent Question ---
>>> procedure TForm1.FormMouseDown [..]
>>> var
>>> Btn: TButton;
>>> begin
>>> Btn := TButton.Create (Self); // What is this doing and why is it
>>> important?
>>>
>>> It's creating a new instance of the TButton class. It's also telling
>>> that new TButton instance that it should set its Owner property to refer
>>> to the current TForm1 instance. (Since the code is running within a
>>> method of the TForm1 class, Self refers to the TForm1 instance on which
>>> that method was called.)
>>>
>>> Descendents of TComponent all have an Owner property. When a component
>>> is destroyed, it will also destroy anything that it owns. That can make
>>> things more convenient since you, the programmer, don't always need to
>>> keep close tabs on every control you create. Instead, you can assign
>>> responsibility to some other control.
>>>
>>> Btn.Parent := Self;
>>> // Why is setting the buttons parent to it self important?
>>>
>>> That's not what it's doing; that would be an invalid operation. It's
>>> setting the new button's Parent property to refer to the current TForm1
>>> instance. The button is now a child control of the form.
>>>
>>> //Can it really be its own parent?
>>>
>>> No. Luckily, that's not what the code is doing.
>>>
>>> //Wouldent the parent be the form the button is on? (procedure TForm1)
>>>
>>> Yes. But not automatically. By default, a control does not have a parent
>>> at all.
>>>
>>> // Why does parent need to be set?
>>>
>>> So the control knows where to display itself.
>>>
>>> Every windowed control in the system needs to have a parent. Windows
>>> without parents are called top-level windows, but they cannot be
>>> controls such as buttons, edit boxes, and tool bars. If a VCL object's
>>> Parent property hasn't been set, then it doesn't have a parent, so when
>>> it comes time for the VCL object to make itself known to the OS by
>>> allocating itself a window handle, there will be problems since the OS
>>> won't know what to do with this orphan window.
>>>
>>> "Use the Parent property to get or set the parent of this control. The
>>> parent of a control is the control that contains the control. For
>>> example,
```

>> > *if an application includes three radio buttons in a group box, the group  
>box  
>> > is the parent of the three radio buttons, and the radio buttons are the  
>> > child controls of the group box." – D7 Help*  
>>  
>> *Uh, yes, thank you. So what?*  
>>  
>> --  
>> *Rob*  
>>  
>  
>