

# Re: reduce redundant code

---

*Source:*

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2008-03/msg01090.html>

---

- *From:* "MP" <NoSpam@xxxxxxxxxx>
  - *Date:* Tue, 11 Mar 2008 14:19:10 -0500
- 

"Ralph" <nt\_consulting64@xxxxxxxxxx> wrote in message  
<news:%239iw4B6gIHA.5160@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>

"MP" <NoSpam@xxxxxxxxxx> wrote in message  
<news:OnSAUgXfiHA.3400@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>

Is there any merit to the following idea?

I find myself repeating lines similar to below in many routines  
the idea being I may want to process all objects of just objects who have  
certain property or whose "container" or "owner" or "child" has certain  
property(ies)

snip

thanks for any comments  
mark

Your actually chewing on several things at once here.  
Get Fowler's book "Refactoring: Improving the Design of Existing Code".  
Impossible to give a specific answer as it depends on the over-all object  
of  
your design.  
Hints: (??? no idea of this is useful or not. <g>)  
Whenever you find yourself with a ton of parameters, think object.  
If only certain objects of a specific type support something, why not it  
let  
the object handle it or ignore it?  
"Filters" are usually only concerned with bags of stuff. Make it a  
property/method of the Bag.

-ralph

Re: reduce redundant code

Hi Ralph,  
re: your quote

"Filters" are usually only concerned with bags of stuff.

in my case the bag of stuff is the api of the external program my utils  
"interact with"  
that bag has lots of kinds of objects

the api presents a collection of all the objects in that file from which I  
want to select

example

appA wants to look at objects containing text data  
(but only if they also match certain secondary lists of properties)

so in my current/older code all my routines have to do something like

For Each obj In Bag

'primary test...object type

'only want objs of certain group of types

If TypeOf obj is x

Or If TypeOf obj is y

Or If TypeOf obj is z then

'obj type ok

'secondary test...other properties of interest

'now only objs w/ certain props are of interest in this case so...

if obj.Propa = a and obj.Propb = b and obj.Propc = c etc then

'do obj

Next

that kind of multiple branches of tests litter my code everywhere  
everytime i write another routine to look at certain things i have to  
rewrite that group of tests with current values..

so what I came up with so far, based on amdrits suggestion is

IFilter (interface to test for object type and list of secondary  
properties)

cTextFilter(Implements IFilter for objects of types that can contain textual  
data)

cTextVal – an object that can contain a textual value

Sub Routine1()

'create textfilter object

Dim oTF as IFilter

Set oTF = New cTextFilter

'set secondary properties

oTf.PropA = a : oTf.PropB= B oTf.PropC= c:

Re: reduce redundant code

## Re: reduce redundant code

'create textvalue object that extracts/updates text value in object  
containing text

```
Dim oTV as cTextVal  
Set oTv = New cTextVal
```

```
'apply current filterObject to TextualObject  
'strategy??? 'aggregation??? 'composition???  
set oTv.ObjectFilter = oTf
```

'now loop becomes

```
For Each obj In Bag  
if oTv.IsValidObject(obj) Then  
'do oTv.SourceObject  
End If  
Next
```

'pseudo code classes  
IFilter class(interface)  
Option Explicit

```
Public Property Get IsValidObject(obj As Object) As Boolean  
'allow implementations test validity of input  
End Property
```

```
'VariousOtherProperties  
Public Property Get PropA() As String  
End Property
```

```
Public Property Get PropB() As String  
End Property  
'etc
```

```
'cTextFilter  
Implements IFilter  
'type of filter for objects which can contain textual data  
Private Property Get IFilter_IsValidObject(obj As Object) As Boolean  
'test for objects that can contain a textual value  
End Property
```

```
'cTextVal  
'object that has textual data  
Property Set ObjectFilter(obj as cTextFilter)  
Set moFilter = obj  
Property Get IsValidObject(obj as object) As Boolean  
If moFilter.IsValidObject(obj) Then  
IsValidObject = True  
Set moSourceObject = obj  
End If  
End Property
```

Re: reduce redundant code

Re: reduce redundant code

Property Get SourceObject()as Object

do I have it all assbackwards?

:~)

Thanks mark

.