

Re: Overrides and mybase ??

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vb/2006-06/msg01848.html>

- *From:* "Jay B. Harlow [MVP – Outlook]" <Jay_Harlow_MVP@xxxxxxxxxxxxxx>
 - *Date:* Sat, 17 Jun 2006 09:52:33 –0500
-

The Bee,

| So you can duplicate a name of a routine many
| times without issue?

Yes this is known as overloading.

| Does compiler than automatically choose which to use
| based on parameters passed?

Yes the compiler looks at the number of parameters & their types to decide
which overload to call.

| What if both routines have same parameters with
| types?

The signatures of the methods (routines) need to be unique on number & type
of parameters. Overloading on return value or parameter name will cause a
compile error.

Advanced information:

..NET 2.0 (VS 2005) allows you to overload generic classes on the number of
type parameters, however you cannot overload a generic simply on the
constraint. For example:

System.Nullable – has shared methods pertaining to all nullable types.

System.Nullable(Of T) – creates a specific nullable type, such as
Nullable(Of Integer) & Nullable(Of Double).

The reason for the two Nullable types is to simplify calling of the methods
on System.Nullable...

—

Hope this helps

Jay B. Harlow [MVP – Outlook]

..NET Application Architect, Enthusiast, & Evangelist

T.S. Bradley – <http://www.tsbradley.net>

Re: Overrides and mybase ??

"The Bee" <TheBee@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
news:OzOmV7ZkGHA.4212@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

| Thank you very much for the time you spent replying. It is greatly
| appreciated.

| So the person who wrote the code really didn't need to do both. They
| could

| have just used the first one? So you can duplicate a name of a routine
| many

| times without issue? Does compiler than automatically choose which to use
| based on parameters passed? What if both routines have same parameters
| with

| types? What happens then?

| Thanks again

"Tom Shelton" <tom@xxxxxxxxxxxx> wrote in message
news:1150497072.093756.179810@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

| >

| > Tom Shelton wrote:

| >> TheBee wrote:

| >>> Hello,

| >>>

| >>> I have seen the following routines in a class and don't understand
| how

| >>> they

| >>> work. For one, why two new() routines? Which is executed? Why not

| >>> just use

| >>> one routine? What does the mybase do in this case? Bit confusing!

| :)

| >>> Thanks

| >>>

| >>> '<remarks/>

| >>> Public Sub New()

| >>> MyBase.New()

| >>> Me.Url =

| >>>

System.Configuration.ConfigurationSettings.AppSettings("ReportServerURL")

| >>> +

| >>> "/ReportService.aspx"

| >>> End Sub

| >>>

| >>> '<remarks/>

| >>> Public Sub New(ByVal ReportServerURL As String)

| >>> MyBase.New()

| >>> Me.Url = ReportServerURL + "/ReportService.aspx"

| >>> End Sub

| >>>

| >>> MyBase.New in this case is not strictly necessary, since it is simply

| >>> calling the base classes default constructor. The only time that

| >>> MyBase.New is required is if the Base class can't be constucted without

Re: Overrides and mybase ??

|>> parameters – in other words, there is no constructor that doesn't take
|>> arguments. You may optionally use it if you want to pass arguments to
|>> the base class constructor... Maybe a small example to make this clear:

```
|>>  
|>> Option Strict On  
|>> Option Explicit On  
|>>  
|>> Imports System  
|>>  
|>> Module Module1  
|>>  
|>> ' Base with only a default constructor  
|>> Private Class Base  
|>> Public Sub New(ByVal param As String)  
|>> Console.WriteLine("Base New – {0}", param)  
|>> End Sub  
|>> End Class
```

|>>
|>
|> Crap... I copied in the wrong constructor.. That should simply be:

```
|>  
|> ' Base with only a default constructor  
|> Public Sub New()  
|> Console.WriteLine("Base New")  
|> End Sub
```

|>
|> Sorry for the confusion!

```
|>  
|> --  
|> Tom Shelton [MVP]
```

```
|>  
|  
|
```