

Forum:[Hack Forums](#)

Thread: [Lazy Type Wrapper, dynamic invoking made easy](#)

Barret - February 22, 2017, 10:56 am

This is a very small yet powerful little invoker that wraps around a type and caches all methods found within that type.

As easy as calling them by name (String) with parameters that don't necessarily have to match, it is forgiven meaning

it will try to convert parameters that can be converted in the signature it requires.

The MethodAttributes parameter is there to filter those methods that are desired, such Shared and Public methods only.

This wrapper is part of a bigger project i'm working on but has many uses for other projects.

Remark: It does not support multiple signatures of the same method

Usage:

PHP Code

```
<?php
```

```
dim Invoker as New Invoker(SomeType, MethodAttributes.Public Or MethodAttributes.Static)
```

```
Dim Result as object = Invoker.Invoke("MyFunction",Arg0,Arg1,Etc...)
```

PHP Code

```
<?php
```

```
Imports System.Reflection
```

```
Public Class Invoker
```

```
    Inherits Dictionary(Of String, MethodInfo)
```

```
    Public Property Type As Type
```

```
    Sub New(Type As Type, MethodAttributes As MethodAttributes)
```

```
        Me.Type = Type
```

```
        For Each m As MethodInfo In Me.Type.GetMethods.Where(Function(x) x.Attributes =
```

```
MethodAttributes)
```

```
            Me.Add(m.Name.ToLower, m)
```

```
    Next
```

```
End Sub
```

```
Public Function Invoke(Name As String, ParamArray Parameters() As Object) As Object
```

```
    If (Me.ContainsKey(Name.ToLower)) Then
```

```
        If (Me.HasSignature(Me(Name.ToLower), Parameters)) Then
```

```
            Return Me(Name.ToLower).Invoke(Me.Type, Parameters)
```

```
        Else
```

```
            Return Me(Name.ToLower).Invoke(Me.Type, Me.TryConvertParamaters(Me(Name.
```

```
ToLower), Parameters))
```

```
        End If
```

```
    End If
```

```
    Return New Object
```

```
End Function
```

```
Private Function TryConvertParamaters(Target As MethodInfo, Parameters() As Object) As
```

```

Object()
    Dim Params As New List(Of Object)
    If (Parameters.Count = Target.GetParameters.Count) Then
        For i As Integer = 0 To Parameters.Count - 1
            If (Parameters(i).GetType IsNot Target.GetParameters(i).ParameterType) Then
                Params.Add(Convert.ChangeType(Parameters(i), Target.GetParameters(i).
ParameterType))
            Else
                Params.Add(Parameters(i))
            End If
        Next
    End If
    Return Params.ToArray
End Function
Private Function HasSignature(Target As MethodInfo, Parameters() As Object) As Boolean
    If (Parameters.Count <> Target.GetParameters.Count) Then Return False
    For i As Integer = 0 To Parameters.Count - 1
        If (Parameters(i).GetType IsNot Target.GetParameters(i).ParameterType) Then
            Return False
        End If
    Next
    Return True
End Function
End Class

```

HFDone - February 22, 2017, 11:36 am

things like this should be in shared class `Invoker.Invoke("MyFunction", Arg0, Arg1, ...)`

Barret - February 22, 2017, 11:46 am

(Feb 22, 2017 12:36 PM)HFDone Wrote: [\[CITE\]](#) things like this should be in shared class `Invoker.Invoke("MyFunction", Arg0, Arg1, ...)`

Wuuuuuuuuuuuuut ? nooooooooooooo way!!! You are soooooooo smart. No seriously, knock it off lad, its annoying. ?

HFDone - February 22, 2017, 12:04 pm

and i thought you learn something ... and changed your behaviour! why you getting mad so quickly you could just ignore it.

soooooooooooooo why its not?

dont expect ppl to reply as you want! ?

INCEPTION_V - February 23, 2017, 9:56 am

(Feb 22, 2017 01:04 PM)HFDone Wrote: [\[CITE\]](#) and i thought you learn something ... and changed your behaviour! why you getting mad so quickly you could just ignore it.

soooooooooooooo why its not?

dont expect ppl to reply as you want! ?

You wot mate ?

Ninja - February 23, 2017, 1:45 pm

This is actually very useful. Thanks for releasing this.

hfTP54 - February 24, 2017, 6:28 am

Was looking for something like this. Thx for the source.

