Execute JavaScript from within a C# assembly

Asked 16 years, 3 months ago Modified 5 years ago Viewed 15k times

21

I'd like to execute JavaScript code from within a C# assembly and have the results of the JavaScript code returned to the calling C# code.

It's easier to define things that I'm not trying to do:

- I'm not trying to call a JavaScript function on a web page from my code behind.
- I'm not trying to load a WebBrowser control.
- I don't want to have the JavaScript perform an AJAX call to a server.

What I want to do is write unit tests in JavaScript and have then unit tests output JSON, even plain text would be fine. Then I want to have a generic C# class/executible that can load the file containing the JS, run the JS unit tests, scrap/load the results, and return a pass/fail with details during a post-build task.

I think it's possible using the old ActiveX ScriptControl, but it seems like there ought to be a .NET way to do this without using SilverLight, the DLR, or anything else that hasn't shipped yet. Anyone have any ideas?

update: From Brad Abrams blog

```
namespace Microsoft.JScript.Vsa
{
    [Obsolete("There is no replacement for this feature. " +
              "Please see the ICodeCompiler documentation for additional help.
" +
              "http://go.microsoft.com/fwlink/?linkid=14202")]
```

Clarification: We have unit tests for our JavaScript functions that are written in JavaScript using the JSUnit framework. Right now during our build process, we have to manually load a web page and click a button to ensure that all of the JavaScript unit tests pass. I'd like to be able to execute the tests during the post-build process when our automated C# unit tests are run and report the success/failure alongside of out C# unit tests and use them as an indicator as to whether or not the build is broken.

c# .net javascript unit-testing codedom

Share

Improve this question

edited Sep 2, 2011 at 8:48

stakx - no longer contributing

asked Sep 15, 2008 at 22:50

ScottKoon

3,493 ● 6 ● 28 ● 29

Follow 84.6k • 20 • 173 • 276

It is such a shame the Javascript implementation for the DLR was abandoned, I believe it was called JScriptX, or this would be been a nice clean integration along with the c# 4.0 dynamic features. – user53791 Sep 12, 2010 at 5:41

6 Answers

Sorted by: Highest score (default)

7

The code should be pretty self explanitory, so I'll just post that.

```
<add assembly="Microsoft.Vsa, Version=8.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/></assemblies>
```

```
using Microsoft.JScript;

public class MyClass {

    public static Microsoft.JScript.Vsa.VsaEngine Engine =
Microsoft.JScript.Vsa.VsaEngine.CreateEngine();

    public static object EvaluateScript(string script)
    {
        object Result = null;
        try
        {
            Result = Microsoft.JScript.Eval.JScriptEvaluate(JScript, Engine);
        }
        catch (Exception ex)
        {
            return ex.Message;
        }

        return Result;
    }

    public void MyMethod() {
        string myscript = ...;
        object myresult = EvaluateScript(myscript);
    }
}
```

Share

Improve this answer

Follow

edited Nov 28, 2019 at 3:47

Wang Liang

4,434 • 7 • 25 • 49

answered Sep 15, 2008 at 22:59

scubabbl

12.8k • 7 • 38 • 37

4 I think someone else has already pointed out that the Microsoft.VSA namespace has been marked obsolete as of VS2008/.NET 3.5... so long as author is aware of this and is targeting

an earlier version I guess it's cool. I should point out I didn't double-check them, though. :)
– Sean Hanley Sep 15, 2008 at 23:09

2

You can use the Microsoft Javascript engine for evaluating JavaScript code from C#

Update: This is obsolete as of VS 2008

Share Improve this answer Follow

edited Sep 15, 2008 at 23:05

answered Sep 15, 2008 at 22:53

Gulzar Nazim
52.2k ● 26 ● 130 ● 170

2 The Microsoft.VSA namespace and all it's objects have been marked as Obsolete as of VS 2008. – ScottKoon Sep 15, 2008 at 23:00

2 @ScottKoon, @Gulzar: So, what should be done instead of using the obsolete code?
– Daniel Daranas May 18, 2011 at 9:20

1

You can run your JSUnit from inside Nant using the JSUnit server, it's written in java and there is not a Nant task but you can run it from the command prompt, the results are logged as XML and you can them integrate them with your build report process. This won't be part of your Nunit result but an extra report. We fail the build if any of those test fails. We are doing exactly that using CC.Net.

Share Improve this answer Follow

answered Sep 16, 2008 at 3:44

user10917
26 ● 2

0

I don't know of any .NET specific way of doing this right now... Well, there's still JScript.NET, but that probably won't be compatible with whatever JS you need to execute :)

Obviously the future would be the .NET JScript implementation for the DLR which is coming... someday (hopefully).

So that probably leaves running the old ActiveX JScript engine, which is certainly possible to do so from .NET (I've done it in the past, though it's a bit on the ugly side!).

Share Improve this answer Follow

answered Sep 15, 2008 at 22:56

tomasr
13.8k ● 3 ● 41 ● 30

If you're not executing the code in the context of a browser, why do the tests need to be written in Javascript? It's hard to understand the bigger picture of what you're trying to accomplish here.

0

Share Improve this answer Follow

answered Sep 15, 2008 at 23:22

Jon Galloway

53.1k ● 25 ● 127 ● 194

Could it be simpler to use JSUnit to write your tests, and then use a WatiN test wrapper to run them through C#, passing or failing based on the JSUnit results?

It is indeed an extra step though.

I believe I read somewhere that an upcoming version of either MBUnit or WatiN will have the functionality built in to process JSUnit test fixtures. If only I could remember where I read that...

0

Share

Improve this answer

Follow

edited Jun 8, 2013 at 14:21

Rob W

349k ● 87 ● 807 ● 682

answered Sep 15, 2008 at 22:56

Sam Wessel

8,848 ● 8 ● 42 ● 44

I've thought about that or using Selenium to run the tests. But it's still an extra step. – ScottKoon Sep 15, 2008 at 23:04

There is some example code regarding using WatiN to run unit tests here: adamesterline.com/2007/05/15/… – Paul Shannon Sep 26, 2008 at 14:17