About

A tool to create a JScript file which loads a .NET v2 assembly from memory.

Releases 5

Packages

README GPL-3.0 license

```
This file is part of DotNetToJScript - A tool to generate a
JScript which bootstraps an arbitrary .NET Assembly and class.
Copyright (C) James Forshaw 2017

DotNetToJScript is free software: you can redistribute it
and/or modify
it under the terms of the GNU General Public License as
published by
the Free Software Foundation, either version 3 of the License,
or
```

```
(at your option) any later version.

DotNetToJScript is distributed in the hope that it will be
useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public
License
along with DotNetToJScript.  If not, see
<http://www.gnu.org/licenses/>.

Usage Notes:

This only works from full trust JScript(obviously), so should
work in
scriptlets etc. By default it will only works if v2/v3/v3.5 is
installed.
However if you specify the '-ver auto' switch when building
the output it
will also work on v4+ only, however that will introduce a
dependency on
WScript.Shell which you might not want.

To use this you'll need to create an assembly which targets
.NET 2 (though
in most cases you can also use 3.5 as you don't tend to see
.NET 2 installed
in isolation. In the assembly implement a class called
TestClass which does
something you want to do in the public, parameterless
constructor.

public class TestClass
{
    public TestClass()
    {
        /* Start notepad */
        Process.Start("notepad.exe");
    }
}

Ensure it's public. Then pass to this tool the path to the
.NET assembly.
If you annotate the class with the ComVisible attribute you
can even interact
with the object after it's created. e.g.

[ComVisible(true)]
```

Contributors 2

tyranid James Forshaw

monoxgas Nick Landers

Languages

- **C#** 100.0%

```
public class TestClass
{
    public void DoSomething(string arg) { }
}

You can change the name of the entry class by using the -c
switch and adding the name.
You can also get the tool to add additional code to interact
with the object by
specifying the -s parameter with the path to a text file
containing the additional
JScript. The created object is named 'o', so for example if
you wanted to call
the DoSomething method load a file containing:

o.DoSomething("SomeArg");

The default mode is to output a JScript file which can be
executed in Windows
Scripting Host. However if you want a scriptlet pass either -m
(for a scriptlet
which can be used from a scriptlet moniker) or -u (for a
scriptlet which can be
used from regsvr32). You can also specify the '-l vba' switch
to output a VBA
file which should work in Office Macros or '-l vbscript' for
VBScript.

Finally by default the tool will output to stdout, you can
output direct to a file
using the -o switch.
```