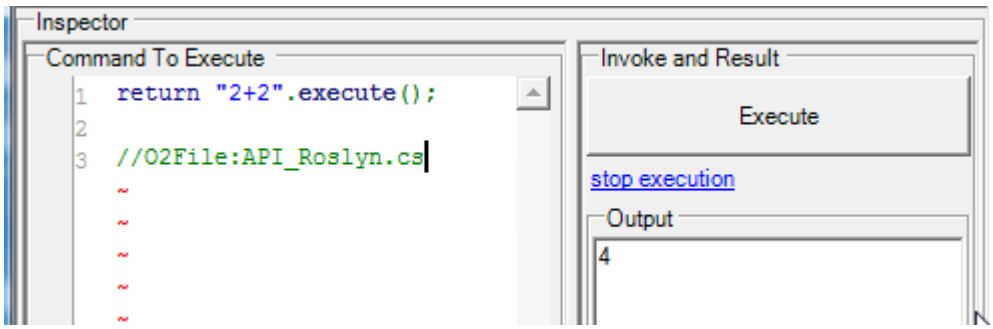
Creating an Roslyn API

Starting with (saved to O2's script folder as API_Roslyn.cs)

```
using System;
using O2.Kernel.ExtensionMethods;
using O2.DotNetWrappers.ExtensionMethods;
using O2.XRules.Database.Utils;

using Roslyn.Scripting.CSharp;
using Roslyn.Scripting;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll

namespace O2.XRules.Database.APIs
{
	public class API_Roslyn
	{

	}

	public static class API_Roslyn_ExtensionMethods_Execute
	{
		public static dynamic execute(this string code)
		{
			return (dynamic)new ScriptEngine().Execute(code);

		}
	}
}
```

we can execute C# scripts via:

```
return "2+2".execute();

//O2File:API_Roslyn.cs
```

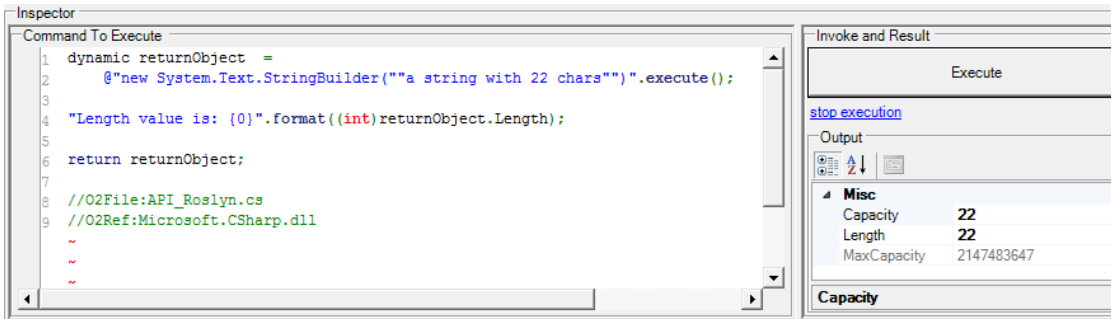


The use of dynamic as return value of execute allow direct access to the returned object values

```
return @"new System.Text.StringBuilder(""a string with 22 chars"")".execute
().Length;

//O2File:API_Roslyn.cs
//O2Ref:Microsoft.CSharp.dll
```

or

```
dynamic returnObject  =
      @"new System.Text.StringBuilder(""a string with 22 chars"")".execute
();

"Length value is: {0}".format((int)returnObject.Length);

return returnObject;

//O2File:API_Roslyn.cs
//O2Ref:Microsoft.CSharp.dll
```



Getting the Syntax tree (either .tree or .parse() will work

new extensionMethods:

```
public static class API_Roslyn_ExtensionMethods_Compiler
{
      public static SyntaxTree tree(this string code)
      {
            return code.parse();
      }

      public static SyntaxTree parse(this string code)
      {
            return SyntaxTree.ParseCompilationUnit(code);
      }
```

code (that uses the Extension methods):

```
var code = @"new System.Text.StringBuilder(""a string with 22 chars"")";

//return code.tree();
return code.parse();

//O2File:API_Roslyn.cs
//O2Ref:Microsoft.CSharp.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

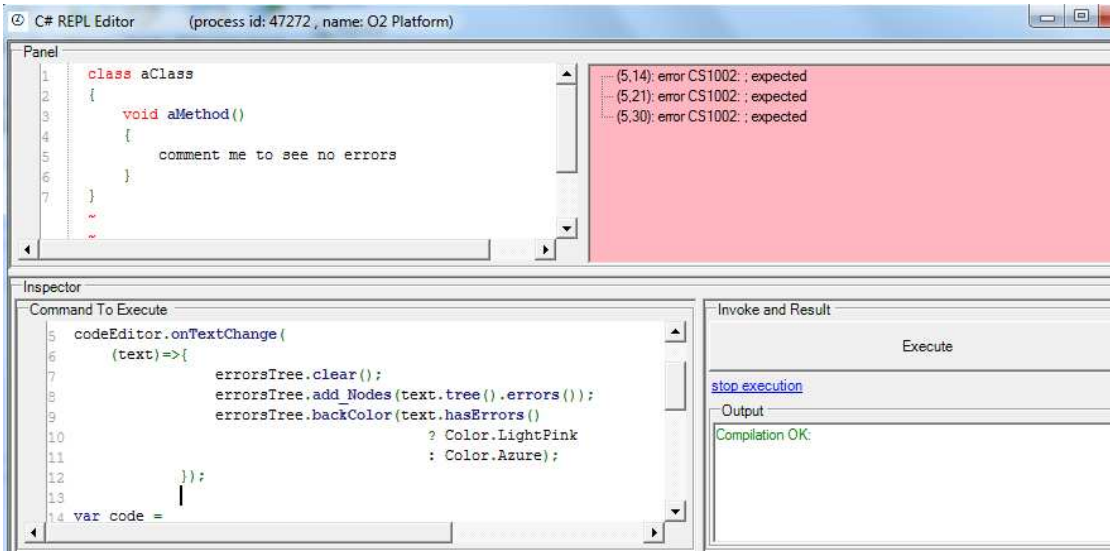
Getting errors

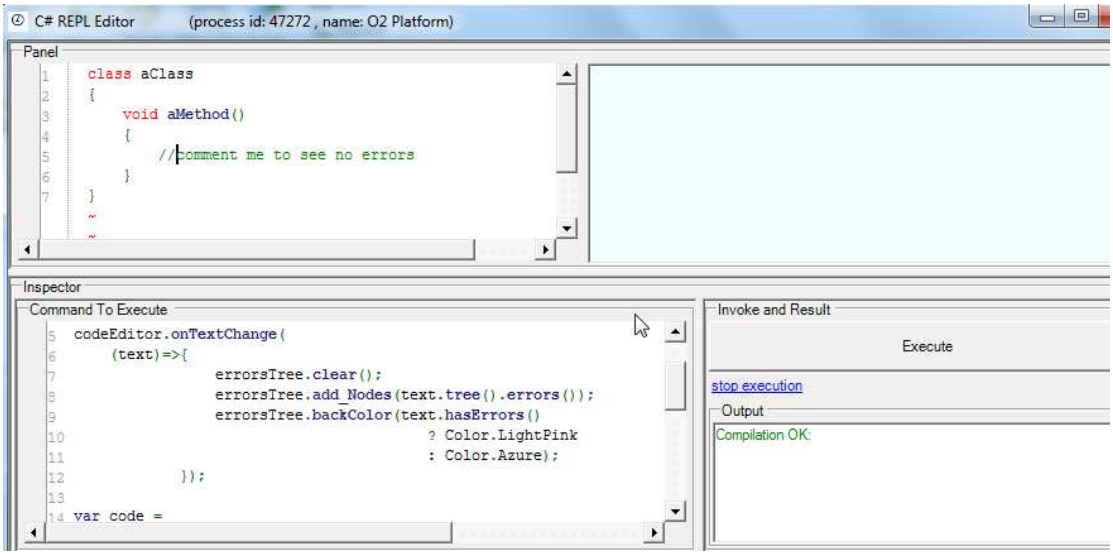
new extension methods:

```
public static List<Diagnostic> errors(this SyntaxTree tree)
{
	return tree.GetDiagnostics().toList();
}

public static bool astOk(this string code)
{
	return code.hasErrors();
}

public static bool hasErrors(this string code)
{
	return code.tree().errors().size() > 0;
}
```

code (that uses the Extension methods):

```
var topPanel = panel.clear().add_Panel();
var codeEditor = topPanel.add_SourceCodeViewer();
var errorsTree = codeEditor.insert_Right().add_TreeView();

codeEditor.onTextChange(
	(text)=>{
			errorsTree.clear();
			errorsTree.add_Nodes(text.tree().errors());
			errorsTree.backColor(text.hasErrors()
								? Color.LightPink
								: Color.Azure);
		  });

var code =
@"class aClass
{
	void aMethod()
	{
		comment me to see no errors
	}
}";
codeEditor.set_Text(code);

//O2File:API_Roslyn.cs
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

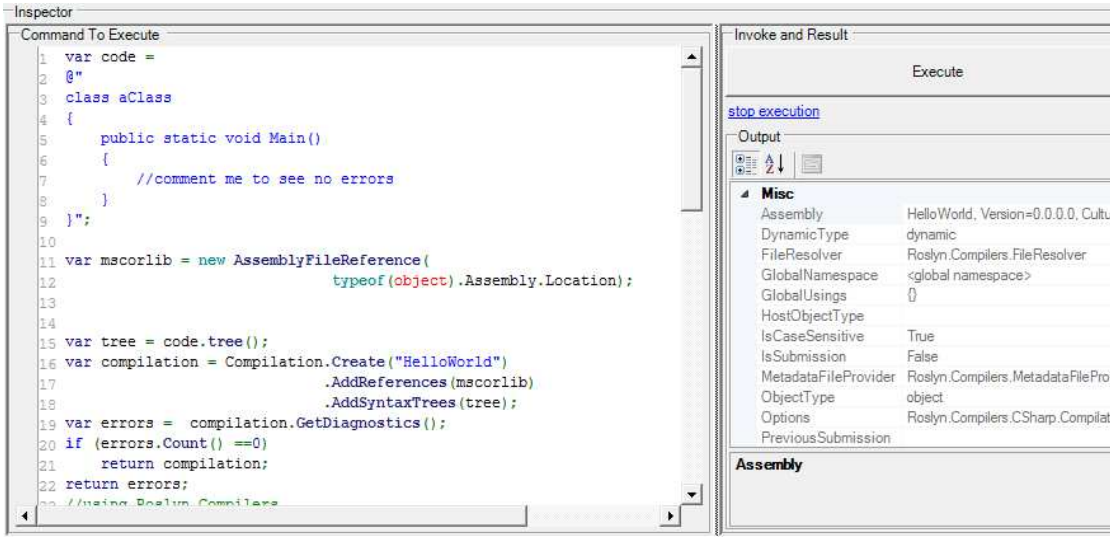




Compiling a script

```
var code =
@"
class aClass
{
       public static void Main()
       {
              //comment me to see no errors
       }
}";

var mscorlib = new AssemblyFileReference(
                                  typeof(object).Assembly.Location);


var tree = code.tree();
var compilation = Compilation.Create("HelloWorld")
                                                .AddReferences(mscorlib)
                                                .AddSyntaxTrees(tree);
```

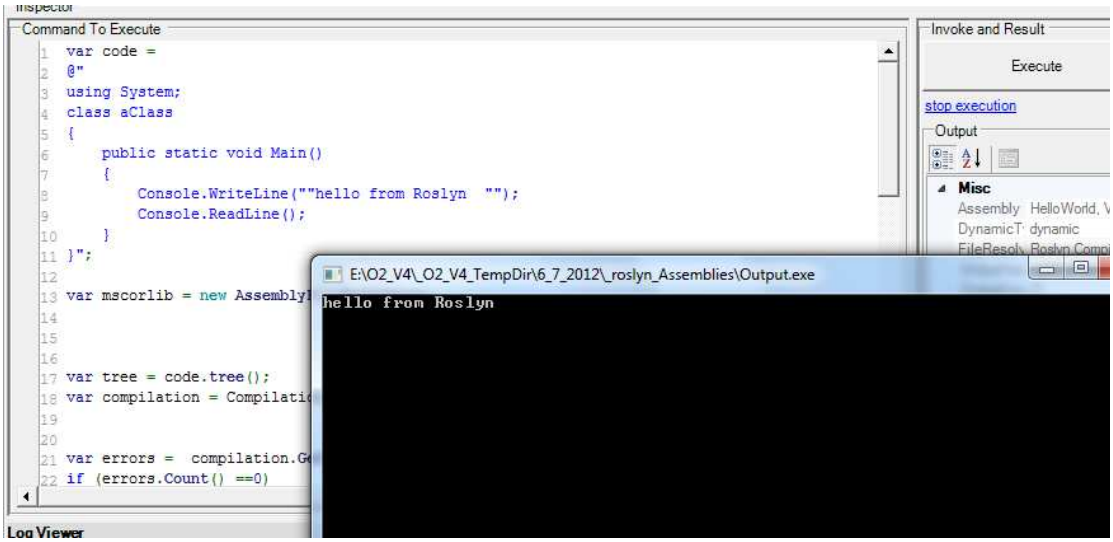
```
if (errors.Count() ==0)
       return compilation;
return errors;

//using Roslyn.Compilers
//using Roslyn.Compilers.CSharp
```



Creating an EXE from code (and pdf and xml comments)

```
var code =
@"
using System;
class aClass
{
       public static void Main()
       {
              Console.WriteLine(""hello from Roslyn ... "");
              Console.ReadLine();
       }
}";

var mscorlib = new AssemblyFileReference(
                                  typeof(object).Assembly.Location);


var tree = code.tree();
var compilation = Compilation.Create("HelloWorld")
                                                .AddReferences(mscorlib)
                                                .AddSyntaxTrees(tree);
var errors =  compilation.GetDiagnostics();
if (errors.Count() ==0)
{
       var tempDir = "_roslyn_Assemblies".tempDir(false);
       var output = new StringBuilder();

    var exeFilename = tempDir.pathCombine("Output.exe");
    var pdbFilename = tempDir.pathCombine("Output.pdb");
    var xmlCommentsFilename = tempDir.pathCombine("Output.xml");
    EmitResult emitResult = null;

    using (var ilStream = new FileStream(exeFilename,
FileMode.OpenOrCreate))
```

```
    using (var pdbStream = new FileStream(pdbFilename,
FileMode.OpenOrCreate))
    using (var xmlCommentsStream = new FileStream(xmlCommentsFilename,
FileMode.OpenOrCreate))
    {
        emitResult = compilation.Emit(ilStream, pdbFilename, pdbStream,
xmlCommentsStream);
    }
    exeFilename.startProcess();

       return compilation;
}
return errors;

//using System.IO
//using System.Text
//using Roslyn.Compilers
//using Roslyn.Compilers.CSharp

 //O2File:API_Roslyn.cs
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```



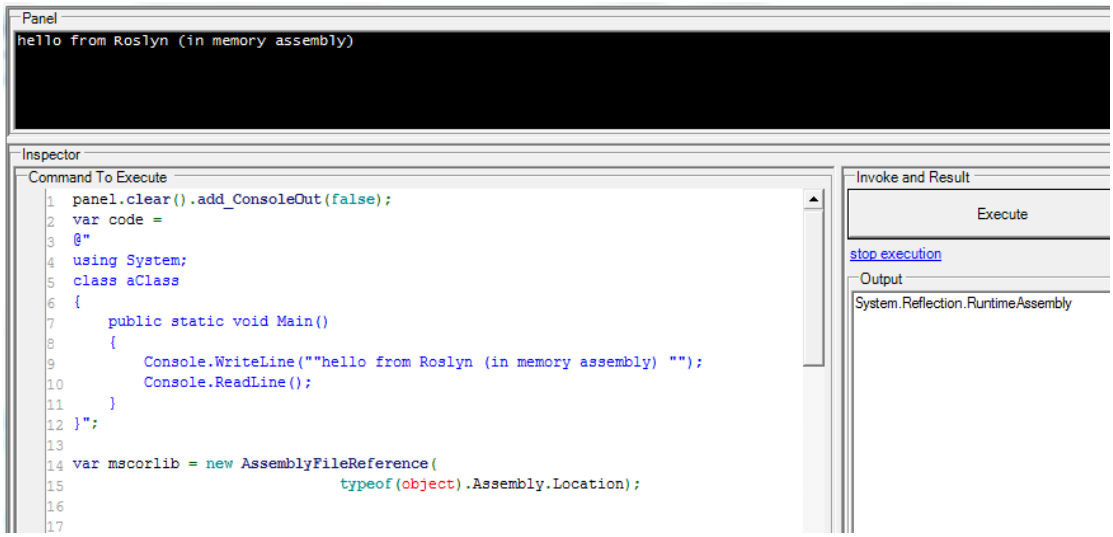
Just creating the exe file:

```
       var tempDir = "_roslyn_Assemblies".tempDir(false);
       var output = new StringBuilder();

    var exeFilename = tempDir.pathCombine("Output.exe");

    using (var ilStream = new FileStream(exeFilename,
FileMode.OpenOrCreate))
    {
        var emitResult = compilation.Emit(ilStream);
    }
    exeFilename.startProcess();
```

Creating and Executing an InMemory Assembly

```
panel.clear().add_ConsoleOut(false);
var code =
@"
using System;
class aClass
{
```

```
        public static void Main()
        {
                Console.WriteLine(""hello from Roslyn (in memory assembly) "");
                Console.ReadLine();
        }
}";

var mscorlib = new AssemblyFileReference(
                                typeof(object).Assembly.Location);


var tree = code.tree();
var compilation = Compilation.Create("HelloWorld")
                                                .AddReferences(mscorlib)
                                                .AddSyntaxTrees(tree);
var errors =  compilation.GetDiagnostics();
if (errors.Count() ==0)
{
    var ilStream = new MemoryStream();
       compilation.Emit(ilStream);
       var assembly = Assembly.Load(ilStream.ToArray());
       assembly.EntryPoint.invokeStatic();

       return assembly.typeFullName();
}
return errors;

//using System.Reflection
//using System.IO
//using Roslyn.Compilers
//using Roslyn.Compilers.CSharp

 //O2File:API_Roslyn.cs
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
//O2File:API_ConsoleOut.cs
```



Creating In-Memory assembly using extension methods

new extension methods:

```
public static Compilation compiler(this SyntaxTree tree, string
assemblyName)
            {
```

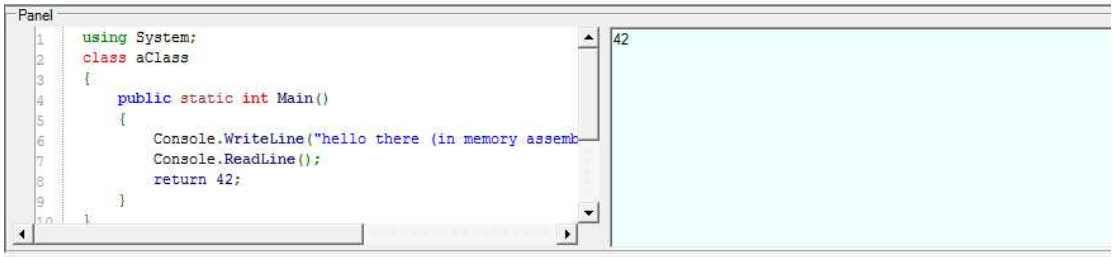
```
                return Compilation.Create(assemblyName)
                                                .AddSyntaxTrees(tree);
        }

        public static Compilation add_Reference(this Compilation
compilation, string assemblyName)
        {
                return compilation.AddReferences
(assemblyName.assemblyReference());
        }

        public static AssemblyFileReference assemblyReference(this
string assemblyName)
        {
                return new AssemblyFileReference
(assemblyName.assembly_Location());
        }

        public static List<Diagnostic> errors(this Compilation
compilation)
        {
                return compilation.GetDiagnostics().toList();
        }

        public static Assembly create_Assembly(this Compilation
compilation)
        {
                var ilStream = new MemoryStream();
                compilation.Emit(ilStream);
                return Assembly.Load(ilStream.ToArray());
        }
```

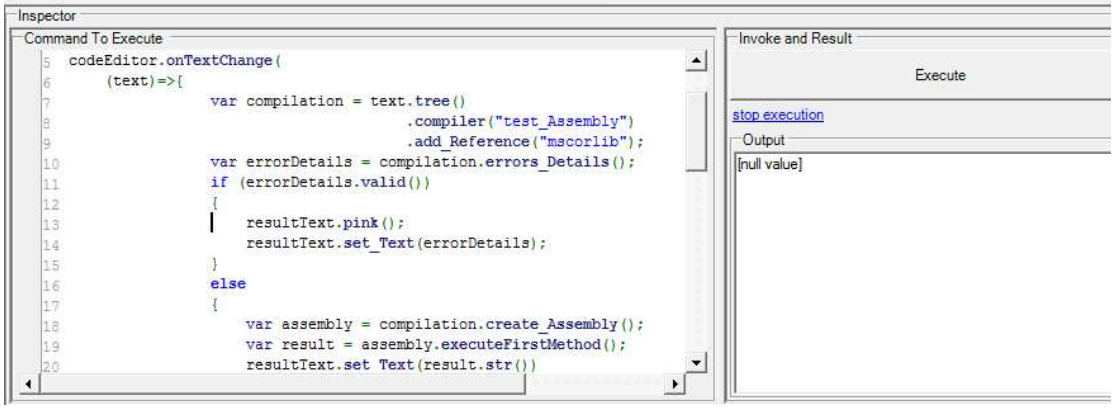
code (that uses the Extension methods):

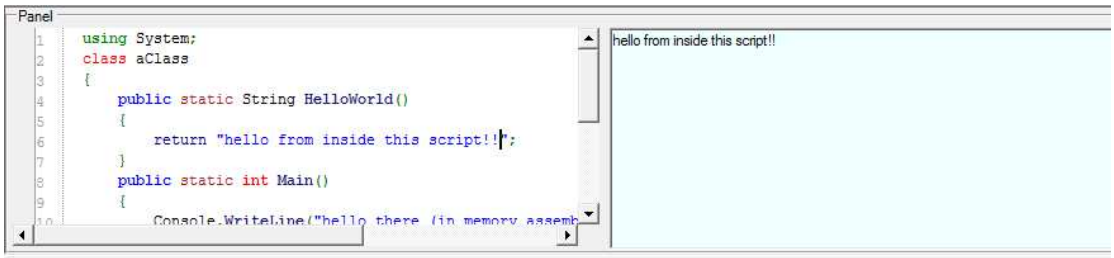
```
panel.clear().add_ConsoleOut(false);
var code =
@"
using System;
class aClass
{
      public static int Main()
      {
            Console.WriteLine(""hello there (in memory assembly) "");
            Console.ReadLine();
            //return ""hello again"";
            return 42;
      }
}";

var assembly = code.tree()
                           .compiler("test_Assembly")
                           .add_Reference("mscorlib")
                           .create_Assembly();

return assembly.EntryPoint.invokeStatic();

//O2File:API_Roslyn.cs
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
//O2File:API_ConsoleOut.cs

//O2Tag_DontAddExtraO2Files
```

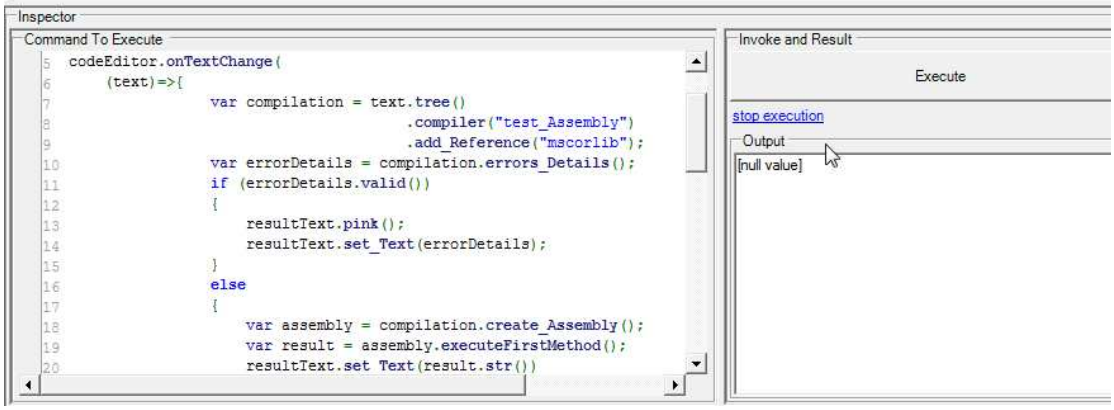
Gui to create and execute script:

```
var topPanel = panel.clear().add_Panel();
var codeEditor = topPanel.add_SourceCodeViewer();
var resultText = codeEditor.insert_Right().add_TextArea();

codeEditor.onTextChange(
       (text)=>{
                           var compilation = text.tree()
                                                                  .compiler
("test_Assembly")
                                                                  .add_Reference
("mscorlib");
                           var errorDetails = compilation.errors_Details();
                           if (errorDetails.valid())
                           {
                                  resultText.pink();
                                  resultText.set_Text(errorDetails);
                           }
                           else
                           {

                                  var assembly = compilation.create_Assembly
();
                                  var result = assembly.executeFirstMethod();
                                  resultText.set_Text(result.str())
                                                .azure();

                           }
                    });
                           //   .create_Assembly();


var code =
@"using System;
class aClass
{
       public static int Main()
       {
              Console.WriteLine(""hello there (in memory assembly) "");
              Console.ReadLine();
              return 42;
       }
}";
codeEditor.set_Text(code);

//O2File:API_Roslyn.cs
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll

//O2Tag_DontAddExtraO2Files
```


Panel
using System;
class aClass
{
public static int Main()
{
Console.WriteLine("hello there (in memory assemb
Console.ReadLine();
return 42;
}
}
42


Inspector
Command To Execute
codeEditor.onTextChange(
(text)=>{
var compilation = text.tree()
.compiler("test_Assembly")
.add_Reference("mscorlib");
var errorDetails = compilation.errors_Details();
if (errorDetails.valid())
{
resultText.pink();
resultText.set_Text(errorDetails);
}
else
{
var assembly = compilation.create_Assembly();
var result = assembly.executeFirstMethod();
resultText.set_Text(result.str())
Invoke and Result
Execute
stop execution
Output
[null value]


Panel
using System;
class aClass
{
public static String HelloWorld()
{
return "hello from inside this script!!";
}
public static int Main()
{
hello from inside this script!!


Inspector
Command To Execute
codeEditor.onTextChange(
(text)=>{
var compilation = text.tree()
.compiler("test_Assembly")
.add_Reference("mscorlib");
var errorDetails = compilation.errors_Details();
if (errorDetails.valid())
{
resultText.pink();
resultText.set_Text(errorDetails);
}
else
{
var assembly = compilation.create_Assembly();
var result = assembly.executeFirstMethod();
resultText.set_Text(result.str())
Invoke and Result
Execute
stop execution
Output
[null value]