# **Скрипты для редактора**

Редактор поддерживает запуск скриптов. Написанных на любом языке, для которого в системе установлен MSIL компилятор. Скорее всего на вашей системе это будет C# или Visual Basic или JScript. Все дальнешие примеры будут приводиться для C#.

## **Hello World!**

| using System.Collections; |
| --- |

| using libdb.Diagnostics; |
| --- |

|  |
| --- |

| **public** **class** MyFirstScript |
| --- |

| { |
| --- |

| **public** IEnumerable Main() |
| --- |

| { |
| --- |

| Log.TraceMessage( "Hello World!" ); |
| --- |

| yield **return** **true**; |
| --- |

| } |
| --- |

| } |
| --- |

Сохраните этот код в какой-нибудь файл, например HelloWorld.cs, и запустите редактор. В редакторе откройте окно со скриптами (View - Show Scripts). Нажмите на кнопку Add Script... ( ), выберите файл со скриптом - после этого скрипт появится в списке доступных скриптов. Теперь можно выделить его и нажать Run Script ( ) (или просто double-click на имени скрипта) и в Output вы увидите Helo World!.

А еще можно запускать скрипт из командной строки при запуске редактора. Для этого надо написать run <имя\_файла\_со\_скриптом>. Запуск скрипта из командной строки не добавляет скрипт в список.

## **Как это работает?!**

Все очень просто: код скрипта компилируется в Assembly, дальше во всех классах из этого Assembly ищется функция **Main** с определенными параметрами, потом эти функции выполняются. Порядок выполнения функций **Main**, когда в скрипте описано несколько классов с этой функцией, не определен.

### **Функция Main**

Функция **Main** может иметь один из следующих вариантов сигнатуры:

| **public** IEnumerable Main(); |
| --- |

| **public** IEnumerable Main( EditorLib.IEditorCommonApplication app ); |
| --- |

| **public** IEnumerable Main( EditorLib.IEditorWinApplication app ); |
| --- |

Функция **Main** ищется в порядке описанном выше, т.е. в приведенном ниже примере:

| **public** **class** DoubleMainFunction |
| --- |

| { |
| --- |

| **public** IEnumerable Main() |
| --- |

| { |
| --- |

| Log.TraceMessage( "IEnumerable DoubleMainFunction.Main()" ); |
| --- |

| yield **return** **true**; |
| --- |

| } |
| --- |

|  |
| --- |

| **public** IEnumerable Main( EditorLib.IEditorCommonApplication app ) |
| --- |

| { |
| --- |

| Log.TraceMessage( "IEnumerable DoubleMainFunction.Main( EditorLib.IEditorCommonApplication app )" ); |
| --- |

| yield **return** **true**; |
| --- |

| } |
| --- |

| } |
| --- |

В *Output* будет выведено *IEnumerable DoubleMainFunction.Main()*

### **yield return true**

Код в функции Main выполняется после загрузки основной формы редактора и в потоке этой формы. Это означает, что проблемы с вызовом функций формы из потока формы - не стоит. Однако при таком подходе возникает проблема - обработать очередь сообщений формы. Конструкция **yield return true** передает управление в форму, но через некоторый интервал времени управление возвращается обратно в **Main**. На данный момент этот интервал равен 25ms.

### **Директива using**

Директива **using** используется как обычно в C#. По умолчанию никаких **using** директив не установлено. Для удобства стоит писать следующий список **using**'ов:

| using System; |
| --- |

| using System.Collections; |
| --- |

| using DBTypes; |
| --- |

| using EditorLib; |
| --- |

| using EditorLib.Extensions; |
| --- |

| using libdb.DB; |
| --- |

| using libdb.Diagnostics; |
| --- |

### **Директива #ref**

Директива **#ref** используется для добавления Assembly, от которых зависит код вашего скрипта. По умолчанию в этот список входят: *mscorlib.dll*, *System.dll*, *System.Windows.Forms.dll*, *libdb.Net.dll*, *WeifenLuo.WinFormsUI.Docking.dll*, *EditorLib.dll*, *Types.DBTypes.dll*. При написании скриптов для конкретного проекта (а пока такой проект только один) стоит писать следующий список **#ref**'ов:

| /\* |
| --- |

| #ref EditorNative.dll |
| --- |

| #ref EditorPlugins.dll |
| --- |

| #ref PF\_Types.DBTypes.dll |
| --- |

| #ref PF\_EditorNative.dll |
| --- |

| #ref PF\_Editor.exe |
| --- |

| \*/ |
| --- |

Обратите внимание: все **#ref** обнесены комментариями - это для того, чтобы редактор C# (например Visual Studio) не ругались на неизвестные директивы. Использование комментария // недопустимо, т.к. ключевое слово **#ref** должно быть первым в строке.

### **Директива #about**

Директива **#about** используется для добавления описания скрипта. В скрипте может быть только 1 директива #about и все описание должно быть написано в одну строку.

| /\* |
| --- |

| #about этот скрипт завоевывает Мир |
| --- |

| \*/ |
| --- |

Обратите внимание: директивы обнесены многострочными комментариями - это для того, чтобы редактор C# (например Visual Studio) не ругались на неизвестные директивы.  
Использование комментария // недопустимо, т.к. строка должна начинаться с директивы.

### **Директива #lang**

Директива **#lang** задает язык программирования на котором написан скрипт. По-умолчанию имеет значение **CSharp**.

## **Как с этим работать?!**

Писать скрипты в блокноте конечно можно, но... неудобно, особенно после того, как привыкнешь к удобствам Visual Studio. Для того, чтобы не отказываться от Visual Studio были созданы .sln/.csproj с прописанными References, и если в этот проект добавить .cs файл со скриптом, то все вещи типа Resolve, Code Complete и т.п. начнут работать под Visual Studio. Да, можно использовать бесплатную Visual Studio C# Express. Для проекта PW эти .sln можно найти в каталоге /Tools/Scripts.

## **Запуск скрипта при запуске редактора**

Можно запустить скрипт при запуске редактора из командной строки. Вот как выглядит вызов редактора в этом случае:

| PF\_Editor.exe run ScriptRunTest.cs -q |
| --- |

Ключик **-q** означает, что надо закрыть редактор после работы скрипта.

## **Tips-n-Tricks**

Здесь будут собираться различные интересные функции с примерами и прочими. Разумеется интересных функции намного больше чем здесь написано, так что смотрите исходники!

### **Сделать скриншот**

| Код | EditorRender.DumpScreenShoot( string fileName ) |
| --- | --- |
| #ref | #ref EditorNative.dll |
| using | using EditorNative |

### **Управление сценой**

Через SceneObjectViewer доступны методы:

* **SetPause( bool )**
* **SetSpeed( int )**:  
  0 - 1FPS  
  1 - 1/16  
  2 - 1/8  
  3 - 1/4  
  4 - 1/2  
  5 - normal  
  6 - 2  
  7 - 4  
  8 - 8  
  9 - 16  
  10 - 0 ( pause )
* **SetFrame( int )** - во фреймах
* **SetTime( float )** - в секундах

| ... |
| --- |

| using EditorPlugins.Scene.Viewers; |
| --- |

| using EditorPlugins.Scene.Viewers.Effect; |
| --- |

|  |
| --- |

| namespace Scripts |
| --- |

| { |
| --- |

| **class** SimpleViewerTest |
| --- |

| { |
| --- |

| **public** IEnumerable Main(EditorLib.IEditorWinApplication app) |
| --- |

| { |
| --- |

| string path = "/Creeps/B/Melee2/\_.SOBJ"; |
| --- |

|  |
| --- |

| DBID id = DBID.FromFileName(path, **false**); |
| --- |

| AbstractEditor editor = app.Application.Extensions.OpenEditor(id, app, **null**, **false**); // Открываем объект |
| --- |

| SceneObjectViewer viewer = (SceneObjectViewer)editor; |
| --- |

| viewer.SetPause(**true**); |
| --- |

| viewer.SetTime(4.9f); |
| --- |

|  |
| --- |

| yield **return** **true**; |
| --- |

| } |
| --- |

| } |
| --- |

| } |
| --- |