

Номинация: **«ОБ ЭТОМ УЖЕ ПИСАЛИ»**

Автор: **Ларс Берман** (*Lars Behrmann*) lb@OpenDocument4all.com

Источник (URL): <http://opendocument4all.com/content/view/68/47/>

Автор перевода: **Чернов Дмитрий**

e-mail: dmchernov@rambler.ru

Программирование для OpenOffice на C# / .net

0 OpenOffice и .net

OpenOffice – свободно доступный офисный пакет, который основан на Sun StarOffice. Это - одна из причин, почему OpenOffice действительно устойчив и полон возможностей, так, что Вы не потеряете ничего, если Вы перейдете от Microsoft Office к OpenOffice. Признание OpenOffice увеличивается день за днём. Таким образом Вы должны дать ему шанс и испытать его. Я думаю, что множество принимающих решения IT специалистов компаний думают о переходе с Microsoft Office на OpenOffice, потому что они смогут сэкономить много денег и начиная с версии 2.0 OpenOffice, там действительно отсутствует потребность в обучении служащих. Обращение с графическим интерфейсом начиная с OpenOffice 2.0 очень напоминает Microsoft Office. Итак, почему они не переходят на OpenOffice? Я предполагаю, что они влюблены в Microsoft Office, потому что они имеют слишком много домашних IT-решений, которые для работы требуют Microsoft Office. Если бы эти решения можно было изменить, чтобы они принимались и Microsoft Office и OpenOffice, я думаю много компаний, перешло бы! Хорошо, это потребует некоторых временных затрат на разработку, а так также денег, чтобы осуществить это. Начиная с версии 2.0 OpenOffice, эта стоимость может быть сведена к минимуму, потому что OpenOffice предложил начиная с этой версии программный интерфейс, основанный на .net. Он представлен CLI сборкой, которая является частью каждой установки OpenOffice. Весьма легко для .net разработчика сослаться на эти сборки и написать приложения, которые общаются и взаимодействуют с OpenOffice. Да, это - все! Нет никакой потребности устанавливать SDK, который при работе будет просто ссылаться на сборки. Это действительно преимущество перед другими языками, такими как C++ или Java.

Даже возможна разработка .net для версий OpenOffice ниже 2.0 при использовании технологии Отражения. Я надеюсь, что это короткое руководство покажет Вам, как просто разработать приложения, которые используют OpenOffice.

Настройка первого проекта в Ms Visual Studio .net 02/03

Требования для следующего шага такие: Вы имеете уже установленный OpenOffice версия которого, соответственно, равняется или больше 2.0, и Вы используете Microsoft Visual Studio .net 2002/2003 как среду для разработки.

1. Откройте вашу IDE и создайте новое “Приложение командной строки”. Это ваш выбор, если Вы используете Visual Basic .net или C# как ваш язык программирования по умолчанию. Этот гид будет всегда использовать C# как язык программирования.
2. Перейдите из окна Visual Studio в окно Explorer. Переместитесь в папку CLI сборок. Вы найдете их в вашем каталоге, где установлен OpenOffice в папке с именем “assembly”. В этой папке Вы должны видеть следующие .net сборки.
CLI сборки: cli_basetypes.dll, cli_cppuhelper.dll, cli_types.dll, cli_ure.dll
Скопируйте все сборки в буфер обмена и возвратитесь в ваш каталог проекта. В вашем каталоге проекта создайте новый каталог и называете его “Resources”.

Теперь, вставьте все сборки в нем. Переключитесь назад к вашему окну Visual Studio, и в проводнике решений щелкните правой кнопкой на новой папке "Resources". Если Вы не видите ее, Вы должны включить отображение всех файлов и справочников.

3. Теперь, Вы готовы ссылаться на CLI сборки в пределах вашего проекта. Щелкните правой кнопкой на "References" и выберите "Add references". В появившемся окне "Reference" выберите "Search" и перейдите к вашему каталогу "Resources". Выделите все CLI сборки и выберите "Open". По крайней мере нажмите ОК в окне "Reference", чтобы добавить CLI сборки к вашим ссылкам. Ваше окно проводника решений должно выглядеть подобно изображенному на .

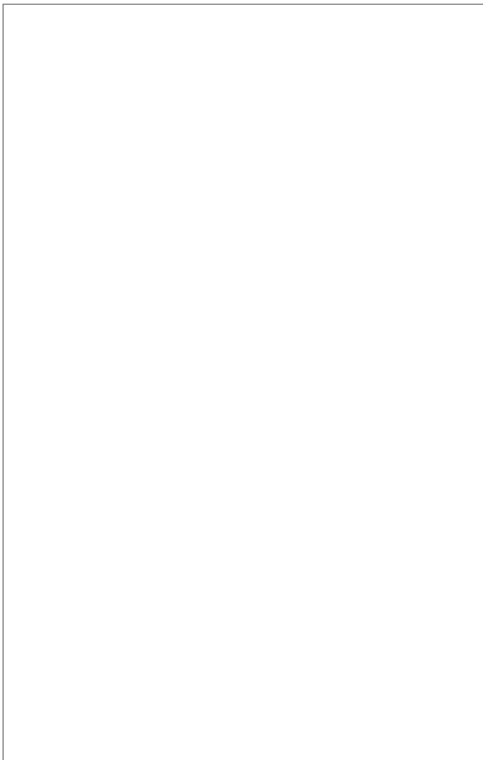


Рис. 1: Проводник решений

Это - все. Теперь, Вы готовы написать ваше первое приложение.

Первое приложение

В качестве первого приложения мы напишем простое приложение, которое соединяется с выполняющимся экземпляром OpenOffice, а если OpenOffice еще не запущен, то запускает его. Если приложение установило связь, мы создаем указатель обслуживания, чтобы общаться с OpenOffice и использовать его для получения доступа к рабочему столу OpenOffice. Если мы получим экземпляр рабочего стола, то мы создадим пустой текстовый документ. В этот пустой документ мы запишем некоторый простой текст, и по крайней мере документ должен быть сохранен.

Исходный текст

Я думаю, что лучший путь, если я покажу Вам весь исходный текст этого приложения. Это действительно не длинно, и я сам помещу говорящие комментарии в коде. Так давайте посмотрим код.

```
using System;
/*
 * Add all needed CLI namespaces to the current class.
 */
using unoidl.com.sun.star.lang;
using unoidl.com.sun.star.uno;
using unoidl.com.sun.star.bridge;
using unoidl.com.sun.star.frame;

namespace ootutorial
{
class OpenOfficeApp
{
    // Определение имени файла. Измените это на существующий путь!
    private static string FileName = @"F:\odtfiles\test.odt";

    [STAThread]
    static void Main(string[] args)
    {
        //Вызов метода bootstrap для получения нового объекта
        //ComponentContext. Если OpenOffice еще не запущен то
        //запустим его и затем возвратим ComponentContext.
        unoidl.com.sun.star.uno.XComponentContext localContext =
            uno.util.Bootstrap.bootstrap();
        //Получение нового диспетчера служб типа MultiServiceFactory,
        //мы нуждаемся в нем, чтобы получить объект desktop и создать
        //новый объект CLI.
        unoidl.com.sun.star.lang.XMultiServiceFactory multiServiceFactory =
            (unoidl.com.sun.star.lang.XMultiServiceFactory)
```



```

/*
 * Use the Reflection namespace to get our Com objects
 */
using System.Reflection;

namespace OpenOfficeAppRef1
{
class OpenOfficeApp
{
    //Определение имени файла. Измените это на существующий путь!
    private static string FileName = @"F:\odtfiles\test.odt";

    [STAThread]
    static void Main(string[] args)
    {
        //Создание нового Type объекта ServiceManager
        Type tServiceManager =
            Type.GetTypeFromProgID("com.sun.star.ServiceManager", true);
        //Создание нового Com объекта ServiceManager используя
        //type объект ServiceManager
        object oServiceManager =
            System.Activator.CreateInstance(tServiceManager);
        //Создание Вашего Com объекта Desktop
        object oDesktop = Invoke(oServiceManager,
            "createinstance",
            BindingFlags.InvokeMethod,
            "com.sun.star.frame.Desktop");
        //Создание массива для загрузки параметров
        Object[] arg = new Object[4];
        arg[0] = "private:factory/swriter";
        arg[1] = "_blank";
        arg[2] = 0;
        arg[3] = new Object[] {};
        //Создание нового чистого документа
        object oComponent = Invoke(oDesktop,
            "loadComponentFromUrl",
            BindingFlags.InvokeMethod,
            arg
        );
        //Создание пустого массива для метода getText
        arg = new Object[0];
        //Получение Com объекта Text
        Object oText = Invoke(oComponent,
            "getText",
            BindingFlags.InvokeMethod,
            arg
        );
        //Запись текста в документ
        Invoke(oComponent,
            "getText",
            BindingFlags.InvokeMethod,
            new Object[1] {"Hello I'm the first text!"}
        );
        //Создание массива для метода storeToUrl
        arg = new Object[2];
        arg[0] = PathConverter(FileName);
        arg[1] = new Object[0] {};
        //Сохранение документа
        Invoke(oComponent,
            "storeToUrl",
            BindingFlags.InvokeMethod,
            arg
        );

        Console.WriteLine("Our first document created via Reflection is done!");
    }

    /// <summary>
    /// Преобразование в формат файла OO
    /// </summary>
    /// <param name="file">Файл.</param>
    /// <returns>Преобразование файла</returns>
    private static string PathConverter(string file)
    {
        try
        {
            file = file.Replace(@"\", "/");
        }
    }
}
}

```

```

    return "file:///"+file;
}
catch(System.Exception ex)
{
    throw ex;
}
}

///  

///  

///  

///  

///  

public static object Invoke(object obj, string method, BindingFlags
binding, params object[] par)
{
    return obj.GetType().InvokeMember(method, binding, null, obj, par);
}
}
}

```

Как Вы можете видеть этот путь программирования OpenOffice также возможен, но я предполагаю, что это надежный путь. Нет никакой информационной поддержки нашим объектам. Вы должны знать каждое свойство и метод. Вы должны использовать его, только если это действительно необходимо.

Дополнительная информация

Если Вы хотите разрабатывать другие приложения, используя OpenOffice, то Вам потребуется дополнительная справочная информация. Для дополнительной справочной информации Вы можете использовать страницы помощи [UNO online](#), которые всегда являются актуальными и [UNO Developers Guide](#). Если Вы загрузите [UNO SDK](#) для OpenOffice, Вы можете просматривать весь этот материал автономно.

Как я упоминал в начале этого документа, нет никакой потребности устанавливать SDK. Это было бы необходимым, только если Вы также планируете использовать такие языки, как Ява или C++. Единственная вещь, в которой Вы нуждаетесь из этой загрузки, – страницы справочной информации. Также Вы должны посетить разделы [www.Oooforum.org](#) – [Code Snippets](#) и [Macro an Api](#), где Вы найдете большое количество тем о .net / C#.