

OpenOffice

programming with C# / .net

A quick start Guide

Author: Lars Behrmann

Email: lb@OpenDocument4all.com

WWW: <http://www.OpenDocument4all.com>

Table of Contents

1. About OpenOffice and .net.....	2
2. Setting up a first project with Ms Visual Studio .net 02/03.....	2
3. A first application.....	3
3.1 The source code.....	3
3.2 The source code using “Reflection”.....	5
4. Additional information.....	7

1. About OpenOffice and .net

OpenOffice is a free available Office Suite which is based on Sun's StarOffice. This is one of the reasons why OpenOffice is really stable and full of features, so that you would miss nothing if you move from Microsoft Office to OpenOffice. The acceptance of OpenOffice is increasing day by day. So you should give it a chance and try it out. I think many IT decision maker of companies think about moving from Microsoft Office to OpenOffice, because they could save a lot of money and since OpenOffice version 2.0 there isn't really a need for a employee training. The Gui handling since OpenOffice 2.0 is very similar to Microsoft Office. So why they don't move to OpenOffice? I guess they stuck on Microsoft Office, because they have too many in house IT solutions which need Microsoft Office to run. If this solutions could changed to accept both Microsoft Office and OpenOffice I think many companies would move! OK, it will cost some development time and so also money to realize this, but it is a one time cost. Since OpenOffice version 2.0 this costs could be downcast to a minimum, because OpenOffice offer since this version a .net based programming interface. Which are represented by the CLI assemblies which are part of each OpenOffice installation. It's quite easy for a .net developer to reference this assemblies and write applications that communicate and interact with OpenOffice. Yes, that is all! There is no need of SDK installation it will run simply by referencing the assemblies. This is a really advantage against other languages like C++ or Java.

Even .net development below OpenOffice version 2.0 is possible by using the Reflection technology. I hope this short guide will show you how you simply develop applications which use OpenOffice.

2. Setting up a first project with Ms Visual Studio .net 02/03

Requirements for this next steps are that you have already a installation of a OpenOffice version greater resp. equal 2.0 and you use Microsoft Visual Studio .net 2002/2003 as development IDE.

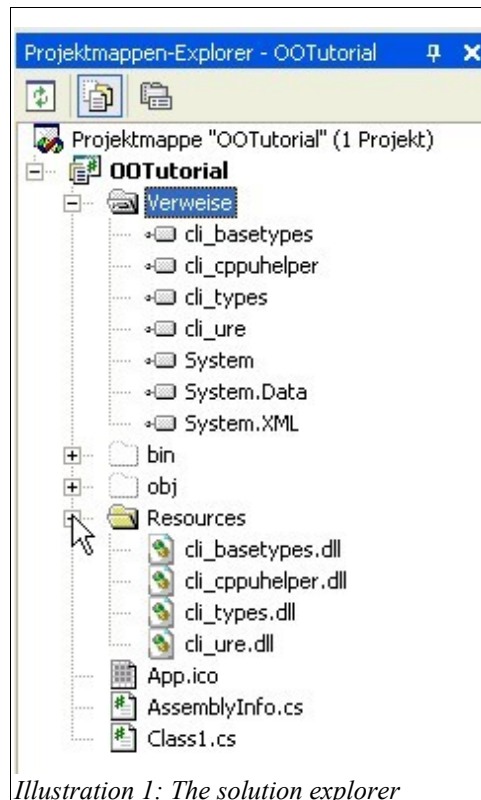
1. Open your IDE and start a new “Command line Application”. It's up to your choice if you use Visual Basic .net or C# as your default programming language. This guide will always use C# as programming language.
2. Change from the Visual Studio window to a Explorer window. Move to the CLI assemblies folder. You will find them in your OpenOffice program installation path in the folder named “assembly”. In this folder you should see the following .net assemblies.

CLI assemblies: cli_basetypes.dll, cli_cppuhelper.dll, cli_types.dll, cli_ure.dll

Copy all assemblies to the clipboard and move back to your project directory. Inside your project directory you create a new folder and name it “Resources”. Now, past all assemblies in there. Switch back to your Visual Studio window and in the Solution Explorer right click

on the folder new folder “Resources”. If you don't see it you have to turn on show all files and directories.

3. Now, you are ready to refernce the CLI assemblies within your project. Right click “References” and choose “Add references”. In the appearing “Reference” window choose “Search” and navigate to your “Resources” folder. Select all CLI assemblies and choose “Open”. At least click OK in the “Reference” window to add the CLI assemblies to your references. Your Solution Explorer window should look similar the next screen shot:



That's all. Now, you are ready to write your first application.

3. A first application

As first application we would programming a simple application which connect to an running OpenOffice instance and if OpenOffice isn't already running it will start a new instance. If the application is connected we create a service handler to communicate with OpenOffice and use it to get access to the OpenOffice desktop. If we receive the desktop instance we will start a blank text document. In this blank document we write some simple text and at least the document should be saved.

3.1 The source code

I think the best way is, if I will show you the whole source code of this application. It isn't really long and I will place self speaking comments inside the code. So let's see the code.

```

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
{
    //Define a file name. Change this to an existing path!
    private static string FileName = @"F:\odtfiles\test.odt";

    [STAThread]
    static void Main(string[] args)
    {
        //Call the bootstrap method to get a new ComponentContext
        //object. If OpenOffice isn't already started this will
        //start it and then return the ComponentContext.
        unoidl.com.sun.star.uno.XComponentContext localContext =
            uno.util.Bootstrap.bootstrap();
        //Get a new service manager of the MultiServiceFactory type
        //we need this to get a desktop object and create new CLI
        //objects.
        unoidl.com.sun.star.lang.XMultiServiceFactory multiServiceFactory =
            (unoidl.com.sun.star.lang.XMultiServiceFactory)
            localContext.getServiceManager();
        //Create a new Desktop instance using our service manager
        //Notice: We cast our desktop object to XComponent loader
        //so that we could load or create new documents.
        XComponentLoader componentLoader
            =
            (XComponentLoader) multiServiceFactory.createInstance(
                "com.sun.star.frame.Desktop" );
        //Create a new blank writer document using our component
        //loader object.
        XComponent xComponent = componentLoader.loadComponentFromURL(
            "private:factory/swriter", //a blank writer document
            "_blank", 0, //into a blank frame use no searchflag
            //use no additional arguments.
            new unoidl.com.sun.star.beans.PropertyValue[0]
        );
        //Cast our component to a the XText interface
        //and write some simple text into document.
        ((unoidl.com.sun.star.text.XTextDocument)xComponent).
            getText().setString("Hello I'm the first text!");
        //After we insert our text, we cast our component to XStorable
        //to save it onto the harddisk
        ((XStorable)xComponent).storeToURL(
            //Convert the file path into a OpenOffice path
            PathConverter(FileName),
            //no additional arguments
            new unoidl.com.sun.star.beans.PropertyValue[0]);
        Console.WriteLine("Your first OpenOffice document is saved!");
        Console.ReadLine();
    }
}

```

```
/// <summary>
/// Convert into OO file format
/// </summary>
/// <param name="file">The file.</param>
/// <returns>The converted file</returns>
private static string PathConverter(string file)
{
    try
    {
        file = file.Replace(@"\", "/");

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

Table 1: Source code of first application

Simple copy this code into the clipboard and replace your class which contain the Main method. Now, you could hit F5 to start the application in Debug mode. It should compile without an error. After the application is started OpenOffice should be started and displaying first a blank writer document into which will immediately written our text. At least the document will be saved. All these steps are so fast that it will look like one single step, but I think that should be no problem ;)

I think was a good starting point for your career as OpenOffice .net developer. Feel free to use this simple application code to experiment with other features of OpenOffice programming.

3.2 The source code using “Reflection”

Now, I will show you how to do the same, but don't using the CLI assemblies. Instead of the CLI assemblies we use the Reflection and Interop technology. This will only quick view on how to develop for versions down OpenOffice 2.0. You will see this isn't nearly as comfortable as the CLI way, but if you have to develop applications for OpenOffice versions 1.1.x.x you will have no other choice. Here comes the code.

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

namespace OpenOfficeAppRefl
{
    class OpenOfficeApp
    {
        //Define a file name. Change this to an existing path!
        private static string FileName = @"F:\odtfiles\test.odt";

        [STAThread]
```

```

static void Main(string[] args)
{
    //Create a new ServiceManager Type object
    Type tServiceManager = Type.GetTypeFromProgID("com.sun.star.ServiceManager",
true);

    //Create a new ServiceManager Com object using our
    //ServiceManager type object
    object oServiceManager = System.Activator.CreateInstance(tServiceManager);
    //Create our Desktop Com object
    object oDesktop = Invoke(oServiceManager,
        "createinstance",
        BindingFlags.InvokeMethod,
        "com.sun.star.frame.Desktop");

    //Create an array for our load parameter
    Object[] arg = new Object[4];
    arg[0] = "private:factory/swriter";
    arg[1] = "_blank";
    arg[2] = 0;
    arg[3] = new Object[] { };
    //Create our new blank document
    object oComponent = Invoke(oDesktop,
        "loadComponentFromUrl",
        BindingFlags.InvokeMethod,
        arg
    );

    //Create an empty array for the getText method
    arg = new Object[0];
    //Get our Text Com object
    Object oText = Invoke(oComponent,
        "getText",
        BindingFlags.InvokeMethod,
        arg
    );

    //Write the text to the document
    Invoke(oComponent,
        "getText",
        BindingFlags.InvokeMethod,
        new Object[1] { "Hello I'm the first text!" }
    );

    //Create an array for the storeToUrl method
    arg = new Object[2];
    arg[0] = PathConverter(FileName);
    arg[1] = new Object[0] { };
    //Save our document
    Invoke(oComponent,
        "storeToUrl",
        BindingFlags.InvokeMethod,
        arg
    );

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

/// <summary>
/// Convert into OO file format
/// </summary>
/// <param name="file">The file.</param>
/// <returns>The converted file</returns>
private static string PathConverter(string file)
{
    try
    {

```

```

        file = file.Replace(@"\", "/");

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

/// <summary>
/// Create a new Com object.
/// </summary>
/// <param name="obj">The object to call invoke on.</param>
/// <param name="method">The methodname to call</param>
/// <param name="binding">One of the BiningFlags Enumertion</param>
/// <param name="par">The params for the invoke method call</param>
/// <returns></returns>
public static object Invoke(object obj,string method, BindingFlags binding,params object[] par)
{
    return obj.GetType().InvokeMember(method,binding,null,obj,par);
}
}
}

```

Table 2: Source code of first application using Reflection

As you can see OpenOffice programming this way is also possible, but I guess it's the hard way. There's no Intellisense support for our objects. You must know every property and method. You should use this only if it's really necessary.

4. Additional information

If you want to develop more applications using OpenOffice you will need further help. For further help you could use the [UNO online](#) help pages which are always up to date and the [UNO Developers Guide](#). If you [download the UNO SDK](#) from OpenOffice you could view all this material also offline.

As I mentioned at the beginning of this document there's no need to install the SDK. This would be only necessary if you also plan to use languages like Java or C++. The only thing that you need of this download are the help pages. Also you should visit the www.Oooforum.org in the [Snippet](#) and the [Macro an Api](#) forum you will find a increasing count of postings about .net / C#.

To be continued ..