

## Introducing Excel-DNA

Excel-DNA is an independent open-source project to integrate .NET into Excel. The Excel-DNA runtime is distributed under a liberal license that explicitly allows commercial use.

Excel supports a number of different programming interfaces. Among these, the Excel C API and XLLs provide the most direct and fastest interface for the addition of high-performance worksheet functions. Unfortunately the C API is not directly accessible from managed code. The Excel-DNA runtime is an integration library that allows managed assemblies such as those created using C# to be integrated with Excel via the C API, providing user-defined worksheet functions, macros and user interface extensions.

The C API allows customization of the function registration, which allows each exported function the set the function wizard category as well as function and argument descriptions. With Excel-DNA these customizations are made by adding .NET attributes to the function declarations in the C# code.

Excel versions '97 through 2010 can be targeted with a single add-in. Various Excel features are supported, including multi-threaded recalculation (Excel 2007 and later), registration-free RTD servers (Excel 2002 and later), customized ribbon interfaces and integrated Custom Task Panes (Excel 2007 and 2010) and offloading UDF computations to a Windows HPC cluster (Excel 2010). Integrating with VBA code is facilitated by allowing classes defined in the managed library to be registered as COM classes and used from VBA code.

One of the advantages of integrating with Excel using the C API and XLLs is that add-ins may be loaded without prior registration, which allows such add-ins to work in reduced privilege environments. Excel-DNA extends this registration-free support to include RTD servers and user-interface extensions like the ribbon interface. A single add-in can thus expose user-defined functions, RTD servers and user interface extensions, and need no administrator privileges, registration or installation steps to function.

The Excel-DNA runtime contains a small loader (the .xll) that loads the .NET runtime, then checks the configuration (.dna) file and accordingly loads the managed assemblies to be registered with Excel. These assemblies are inspected using the .NET reflection mechanism, and the appropriate methods are registered with Excel (with custom information as set by the attributes). Ribbon or command bar interfaces are loaded, and RTD servers registered. Once the reflection-based inspection process and registration is complete, the resulting function exports are directly accessible from Excel, leaving very low function-call overhead at runtime.

A basic add-in will consist of three files:

- The managed assembly containing the functions to be exported, called MyLibrary.dll.
- The Excel-DNA runtime library (distributed as exceldna.xll), renamed to FirstAddIn.xll.
- The configuration file (a text-based xml file), renamed to FirstAddIn.dna, containing the configuration:

```
<DnaLibrary Name="My First AddIn" RuntimeVersion="v4.0">  
  <ExternalLibrary Path="MyLibrary.dll" />  
</DnaLibrary>
```

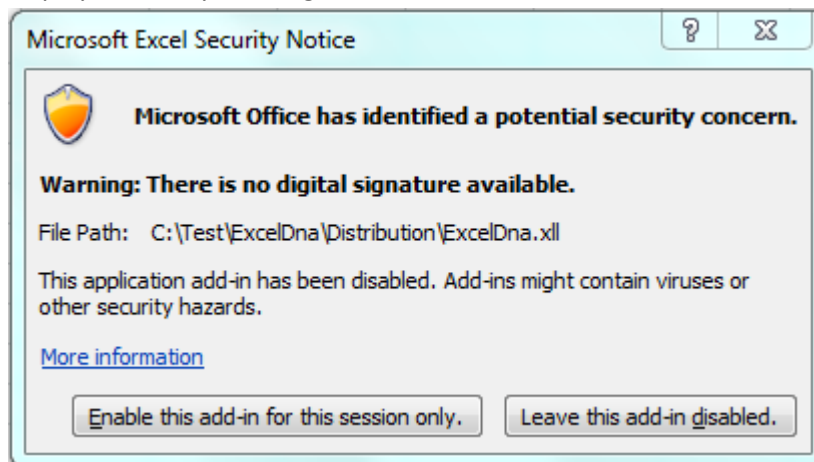
A packing feature allows the Excel-DNA runtime library, configuration files, user add-in assemblies and other dependent libraries to be packed into a single .xll file. The only external requirement for using such an add-in would be that the appropriate .NET runtime version should be installed.

## Step-by-step instructions

### Getting and testing Excel-DNA

In this section we download Excel-DNA and check that the simplest sample add-in works. This ensures that the .NET framework is correctly registered, that the Excel security settings will not prevent the add-ins from loading.

1. Download the latest version of Excel-DNA from <http://exceldna.codeplex.com> and extract the .zip file to a convenient location, say C:\Test\ExcelDna\.
2. Extracting the distribution will create folders called Distribution and Source.
3. In the Distribution folder, double-click the ExcelDna.xll file – Excel should open and possibly display a security warning:



4. Click to enable the add-in in Excel.
5. The add-in registers a function called “AddThem”, which can be entered into a cell :



6. The code creating defining this add-in can be found in the ExcelDna.dna file:

```
<DnaLibrary>
<![CDATA[

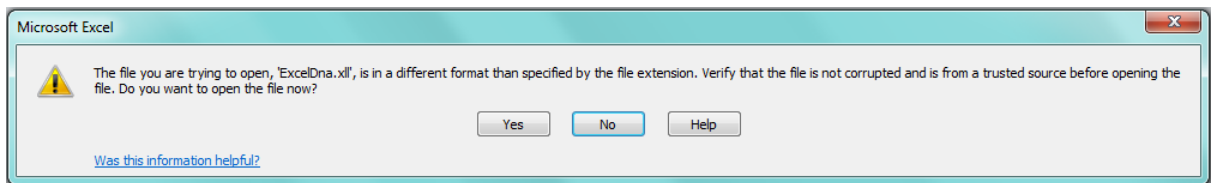
    Public Module Module1
        Function AddThem(x, y)
            AddThem = x + y
        End Function
    End Module

]]>
</DnaLibrary>
```

### Troubleshooting

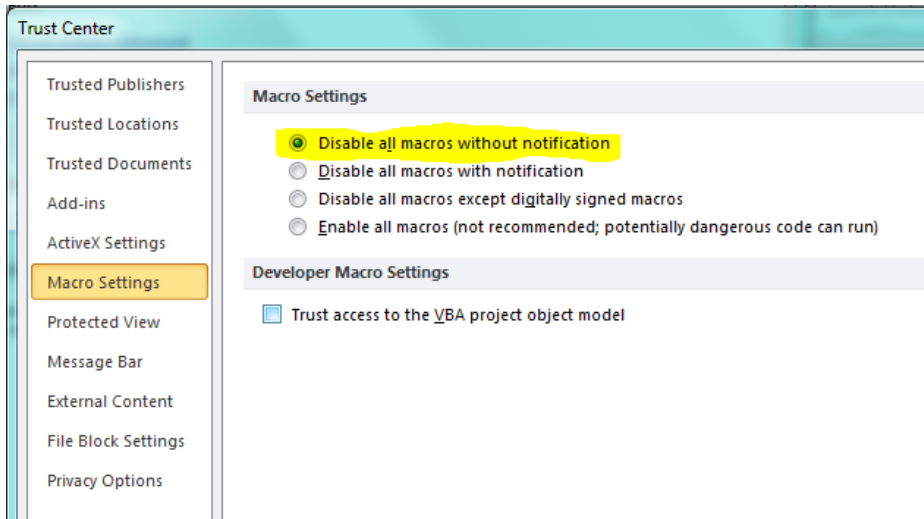
The following are some common problems which may arise when loading the simplest samples add-in:

- Excel claims that the file is in an incorrect format:



This may occur if the 64-bit version of Excel 2010 is installed. Load the corresponding ExcelDna64.xll file instead.

- The .NET runtime is not present.
- Security restrictions prevent Excel from loading the file. The following setting would cause Excel to silently disable all add-ins:

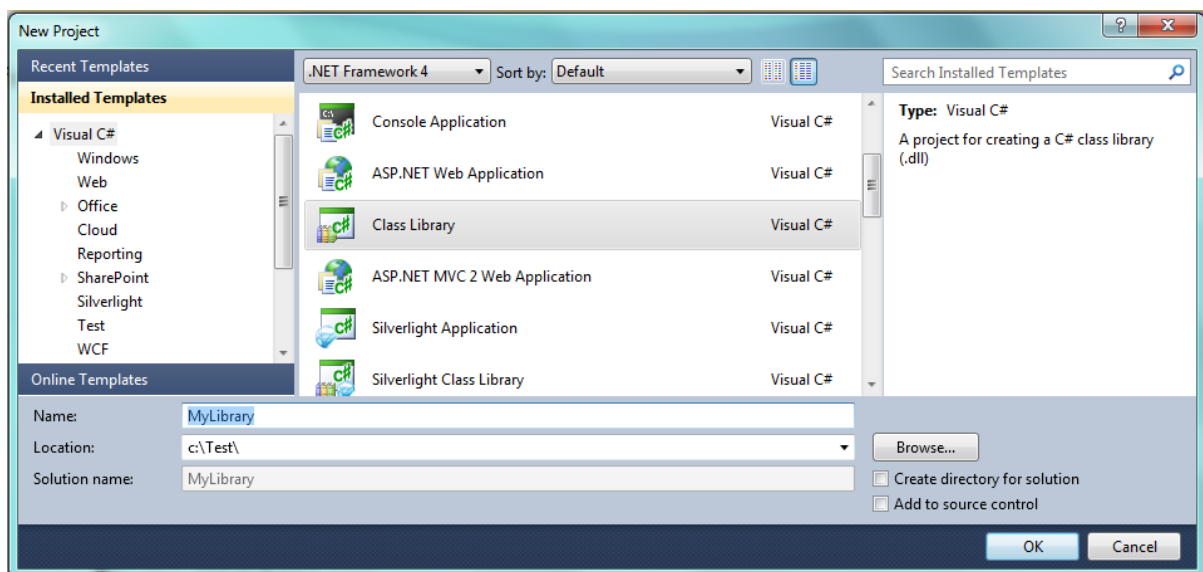


- Some evaluation versions of Excel do not support add-ins; installing Excel without support for the VBA environment also prevents .xll add-ins from loading.

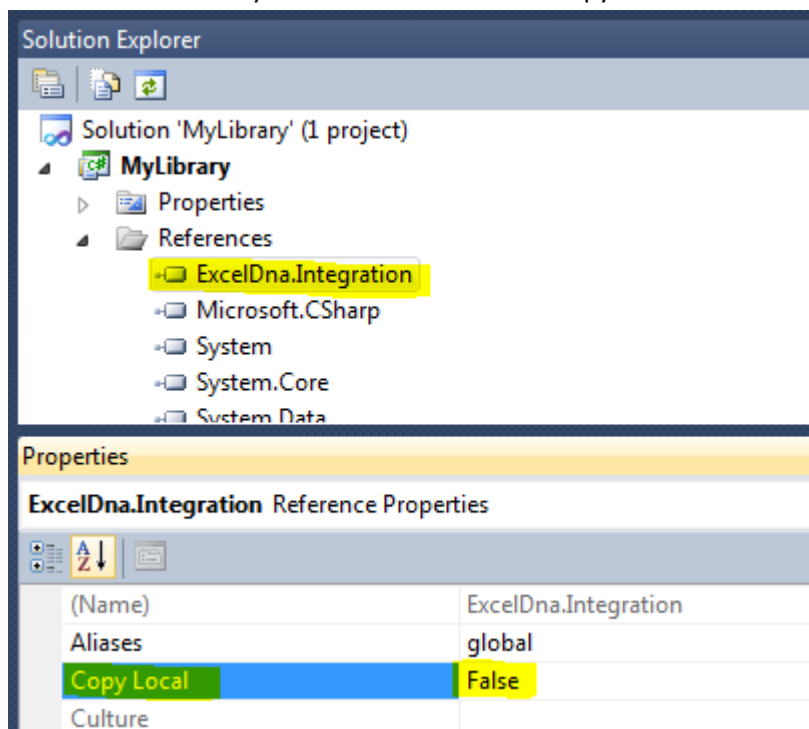
## Creating a new add-in library in C#

In this section we create a simple C# add-in library, and expose it to Excel with the Excel-DNA runtime.

1. In Visual Studio, create a new Class Library project



2. Add a reference to the ExcelDna.Integration.dll assembly (which is in the Excel-DNA Distribution directory). Mark this reference as Copy Local – False:



3. In the Class1.cs file, add a namespace declaration and a public static function to the Class1 class:

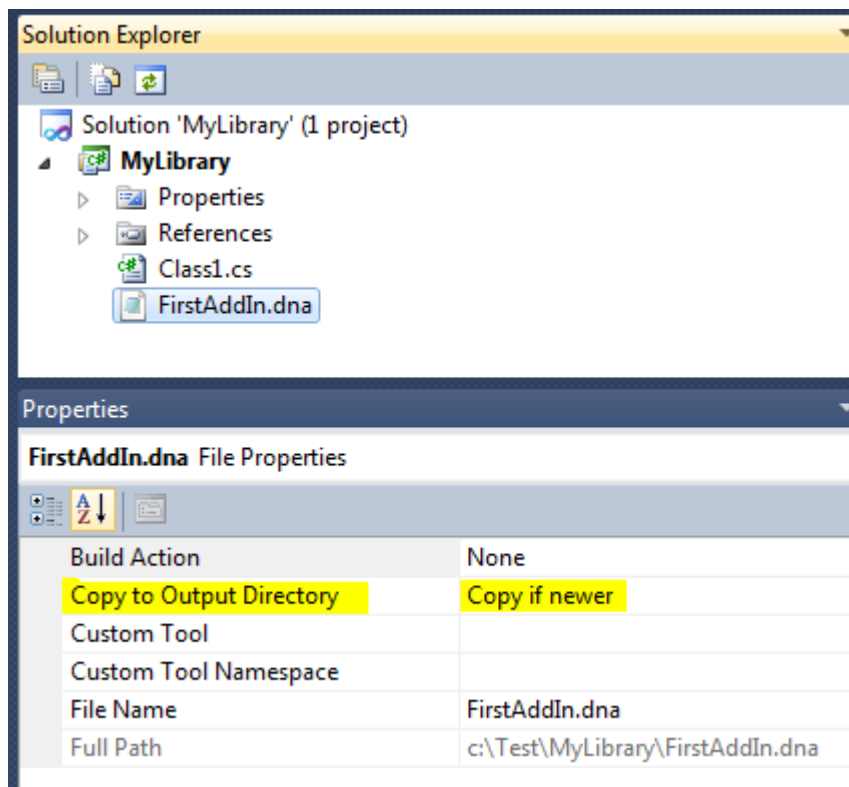
```
using System;
using ExcelDna.Integration;

namespace MyLibrary
{
    public class Class1
    {
        [ExcelFunction(Description="My first Excel-DNA function")]
        public static string MyFirstFunction(string name)
        {
            return "Hello " + name;
        }
    }
}
```

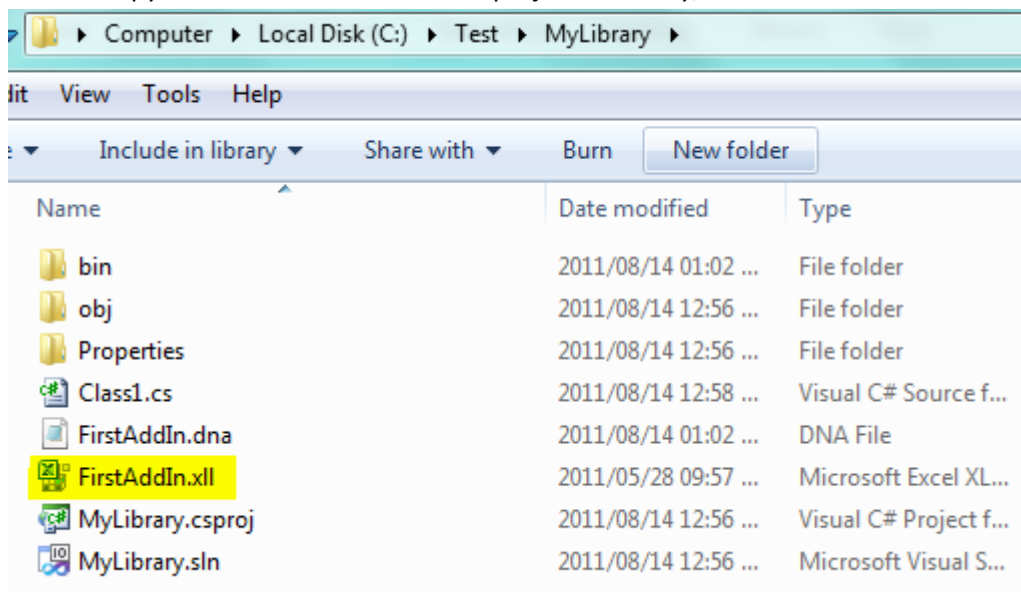
4. Add a file to the project called "FirstAddIn.dna" with the following content:

```
<DnaLibrary Name="First Add-In" RuntimeVersion="v4.0">
  <ExternalLibrary Path="MyLibrary.dll" />
</DnaLibrary>
```

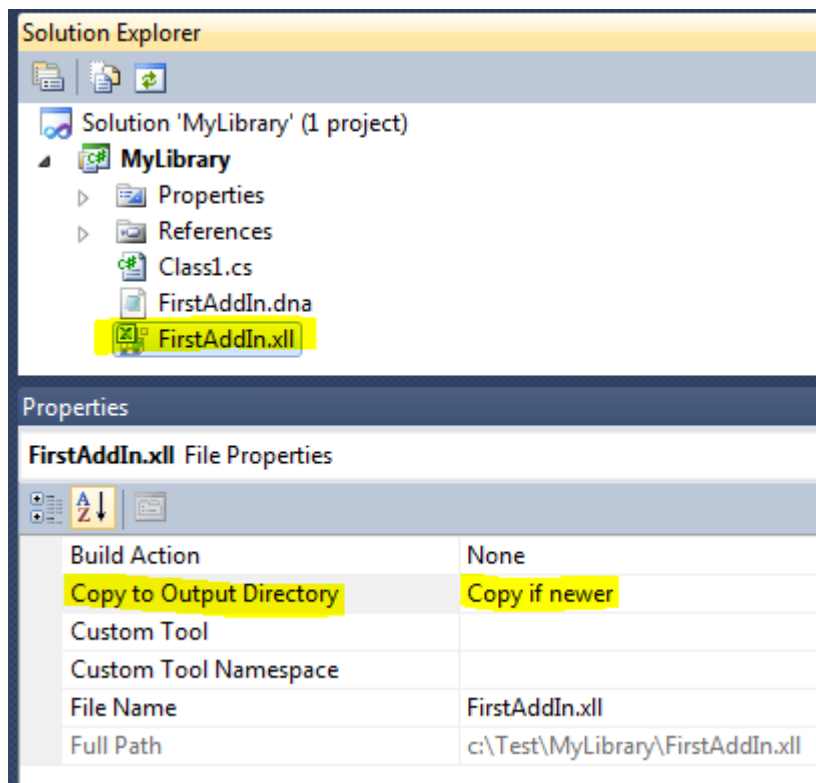
5. In the properties for this file, set that this file will be copied to the Output directory:



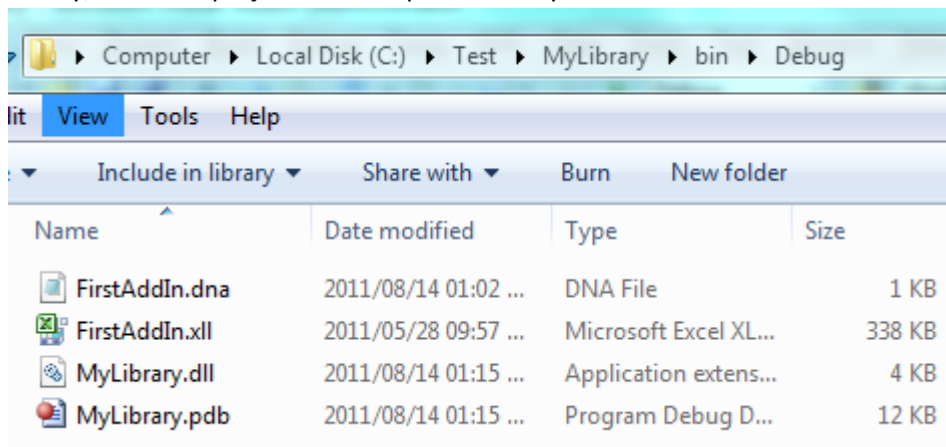
6. Make a copy of the ExcelDna.xll into the project directory, and rename it to FirstAddIn.xll:



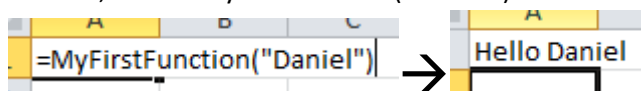
7. Add this file to the project, and in the properties for the file ensure that it will be copied to the Output directory:



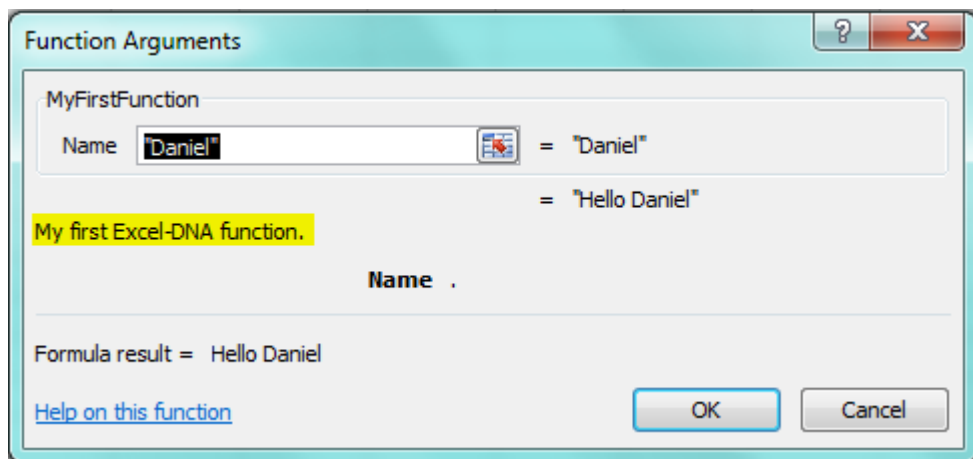
8. Finally, build the project. The Output directory should look like this:



9. Double-click the FirstAddIn.xll to open in Excel (or select File->Open from an Excel session).  
 10. In a cell, enter =MyFirstFunction("Daniel")



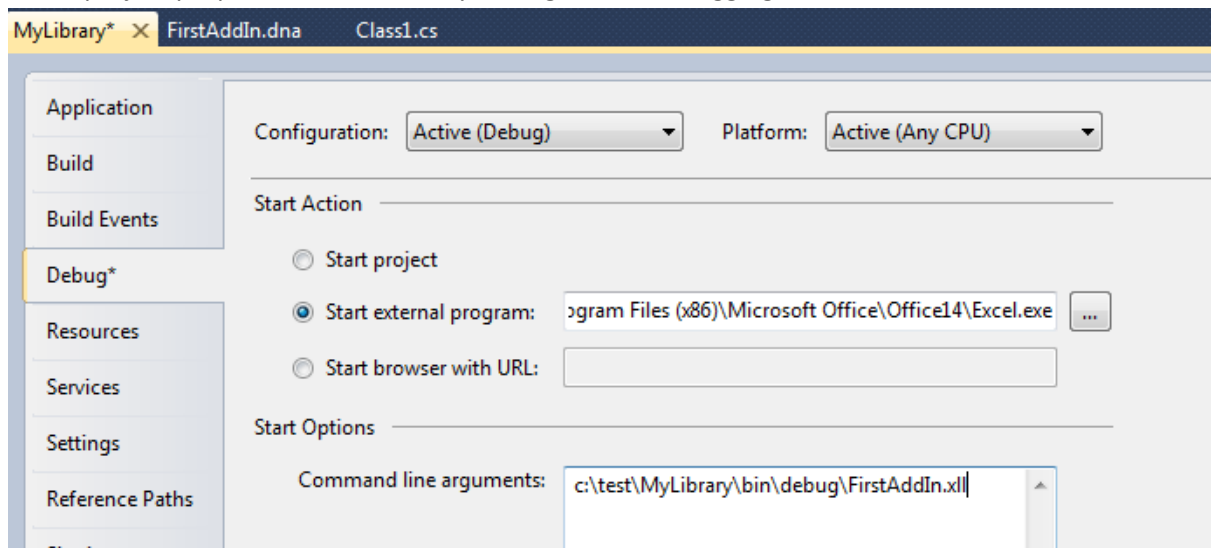
11. Check the function description in the function wizard:



## Debugging the function

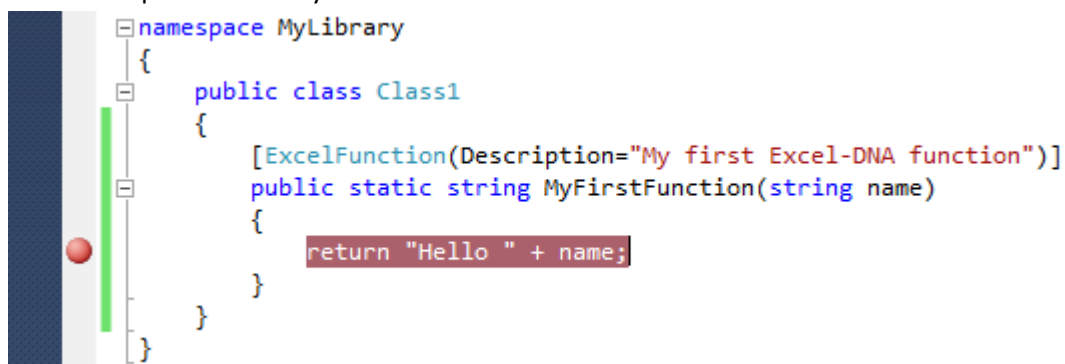
In this section we configure the project to enable debugging of our library.

1. In the project properties for the library, configure the debugging start action as follows:



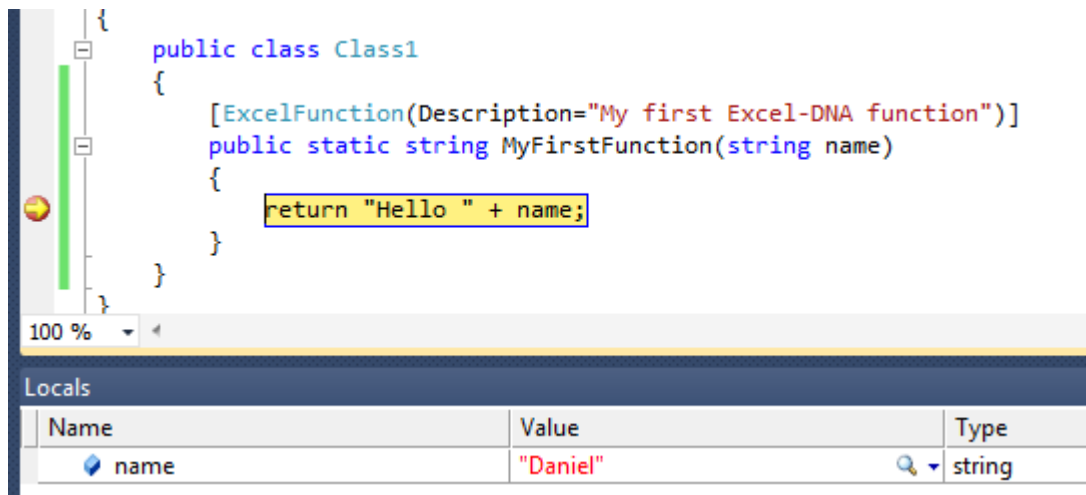
(‘Office14’ is the directory for Office 2010, substitute ‘Office12’ for Office 2007, or another version as appropriate.)

2. Set a breakpoint in the MyFirstFunction function:



3. Press F5 to start debugging – Excel should start and load the add-in. Add a new sheet by selecting File->New or pressing Ctrl+N.
4. Enter the function in Excel and wait for the breakpoint to be hit:





## Packing the add-in for distribution

In this section the add-in is prepared for distribution as a single-file .xll.

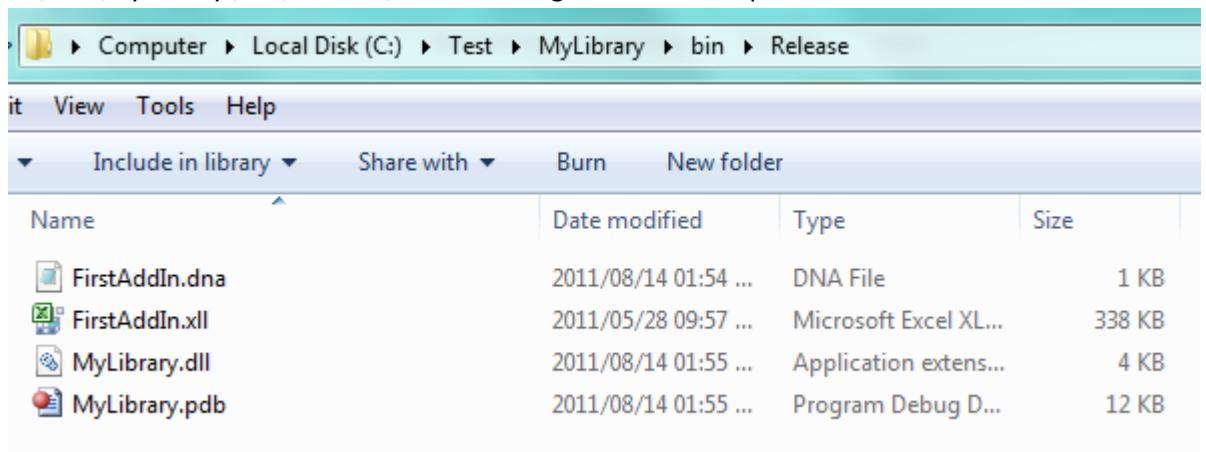
1. Add a packing directive to the FirstAddIn.dna file. This indicates that MyLibrary.dll should be packed into the .xll file.

```

<DnaLibrary Name="First Add-In" RuntimeVersion="v4.0">
  <ExternalLibrary Path="MyLibrary.dll" Pack="true"/>
</DnaLibrary>

```

2. Rebuild the project as a release build.
3. Open a command prompt in the output directory of the project, in this case C:\Test\MyLibrary\bin\Release\. The following files should be present:



4. From the command prompt, run "C:\Test\ExcelDna\Distribution\ExcelDnaPack.exe FirstAddIn.dna". The following output results:

```

C:\Test\MyLibrary\bin\Release>C:\Test\ExcelDna\Distribution\ExcelDnaPack.exe FirstAddIn.dna
Using base add-in FirstAddIn.xll
-> Updating resource: Type: ASSEMBLY_LZMA, Name: EXCELDNA.INTEGRATION, Length: 43546
-> ExternalLibrary path MyLibrary.dll resolved to C:\Test\MyLibrary\bin\Release\MyLibrary.dll.
-> Updating resource: Type: ASSEMBLY_LZMA, Name: MYLIBRARY, Length: 1514
-> Updating resource: Type: DNA, Name: __MAIN__, Length: 385
Completed Packing FirstAddIn-packed.xll.
C:\Test\MyLibrary\bin\Release>

```

5. The result is a file called FirstAddIn-packed.xll which is a single file add-in which can be renamed, distributed and run with no other files required. (Both FirstAddIn.dna and MyLibrary.dll are packed inside the .xll as resources, and will be loaded at runtime.)

## Further development

Some directions for further development of the add-in:

- Accessing the C API and the COM object model from Excel-DNA.
- Multithreading functions, volatile functions, reference parameters.
- Macros, menus, ribbons and custom task panes.
- Registration-free RTD servers.