# How to create your own DLL (With Visual Studio .NET)

#### Introduction

An export DLL is a dynamically linked library, which has functions, classes, and variables that are available to any application that links to this DLL. Suppose that you create a DLL called MyDLL.dll. Then the application, which uses that DLL, needs to include the following files in its project directory:

- MyDLL.h the DLL's header file.
- MyDLL.lib short static library of the DLL.
- MyDLL.dll the compiled DLL.

The actual code resides within the DLL, and will not be linked to the application at compile time (only at run time), thus the application is smaller in size, and can be updated very easily by replacing the DLL.

### **Creating your first DLL**

To create your DLL, follow these steps:

- 1. Stat a new project in Visual Studio .NET.
- 2. Select Visual C++ Projects Win32 and Win32 Console Project.
- 3. Give the project a name (like MyDLL) and press OK.

Project Types:       I         Visual Basic Projects       Visual C# Projects         Visual J# Projects       Visual J# Projects         Visual C++ Projects       Visual C++ Projects         ATL       MFC         MFC       Visual General         Visual General       Visual Projects         Merc       Visual Visual C++         MFC       Visual Visual C++         Min32       Visual Visual C++         Min32       Visual Visual Visual C++         Min32       Visual Visual C++         Visual C++       Visual Visual C++         MFC       Visual Visu	emplates:
Visual Basic Projects         Visual C# Projects         Visual J# Projects         Visual C++ Projects         Image: NET         Image: MyDLL         Location:	
A Console application type of Win32 project.           Mame:         MyDLL           ocation:         C:\Documents and Settings\S	/in32 Console Win32 Project Project
ecation: C:\Documents and Settings\S	
ocation: C:\Documents and Settings\S	
	ai\Desktop 🗾 Browse
Project will be created at C:\Documents and Settings\S	nai\Desktop\MyDLL.

 Click on Application Settings, select DLL and Export Symbols and press Finish.

n 32 Application Witzard pplication Settings Specify the type of applicatio want supported.	n you will build with this project and the	options or libraries you	
Overview	Application type: <u>     Windows application</u>	Add support for:	
	DLL     Static library     Additional options:     Empty project     Export symbols     Precompiled header		
		Finish Cancel	Help

5. That's it; you're done creating the DLL.

#### **Some Explanations**

Before we move to the point of actually doing something with the DLL, here are some explanations about the created project.

Look at the 4 files that were created:

- MyDLL.h header for the DLL
- MyDLL.cpp Implementation of the DLL.
- StdAfx.h Standard include (you can write here the standard #includes like stdlib.h and stdio.h).
- StdAfx.cpp Standard object file **Do not edit this one!**

Both StdAfx files are not interesting, so let's skip them for now.

#### <u>MyDll.h</u>

};

The code below was added by the DLL creation wizard.

```
#ifdef MYDLL_EXPORTS
#define MYDLL_API __declspec(dllexport)
#else
#define MYDLL_API __declspec(dllimport)
#endif
```

// This class is exported from the MyDLL.dll
class MYDLL\_API CMyDLL {
 public:
 CMyDLL(void);

// TODO: add your methods here.

extern MYDLL\_API int nMyDLL;

MYDLL\_API int fnMyDLL(void);

Pre-processor definitions. No need to touch that piece of code. If you decide to change MYDLL\_API to

another constant, make sure you change it references in the code.

This is a class defined in the DLL. To add more methods, simply modify the class.

Note: when implementing the methods, remember to prefix them with MYDLL\_API.

These are a variable and a function exported from the DLL. I will not discuss them here.

Make sure you do this:

- Define MYDLL\_EXPORTS in the project's preprocessor settings.
  - Right-click you project and Properties.
  - $\circ$  Go to Configuration Properties C/C++ and Preprocessor.
  - Make sure that MYDLL\_EXPORTS appear in Preprocessor Definitions line.

DEL Property Pages			
onfiguration: Active(Debug)	Platform: Active(Wir	32) Configuration M	lanager
Configuration Propertie	Preprocessor Definitions	WIN32;_DEBUG;_WINDOWS;_USRD	
General	Ignore Standard Include Path	No	
Debugging	Generate Preprocessed File	No	
🔄 C/C++	Keep Comments	No	
<ul> <li>Preprocessor</li> <li>Code Generatio</li> <li>Language</li> <li>Precompiled He.</li> <li>Output Files</li> <li>Browse Informa</li> <li>Advanced</li> <li>Command Line</li> <li>Linker</li> <li>Preverse Information</li> </ul>			
Build Events     Custom Build Step	Preprocessor Definitions	definer (Ppferene)	
Web Deployment	opecines one or more preprocessor	dennes. (Jo[macro])	
			Help

#### MyDLL.cpp

This code was added by the wizard.

#include "stdafx.h" #include "MyDLL.h"

CPP header. Includes, etc.



The DLL entry point. Look in: <u>http://msdn.microsoft.com/library/def</u> <u>ault.asp?url=/library/en-</u> <u>us/dllproc/base/dllmain.asp</u> for more explanations.

// This is an example of an exported variable MYDLL\_API int nMyDLL=0;

// This is an example of an exported function.
MYDLL\_API int fnMyDLL(void) {
 return 42;

These are the variable and function implementations.

```
// This is the constructor of a class that has been exported.
// see MyDLL.h for the class definition
CMyDLL::CMyDLL() {
    return;
}
```

This is the constructor's implementation.

## **Compile your code**

}

I created another project in the Visual Studio's solution and called it

TestDLL. What we want to do now is use MyDLL in this project. In order to

do it, we need to set some settings.

- 1. Copy MyDLL.h, MyDLL.lib and MyDLL.dll to TestDLL's project directory.
- 2. Include MyDLL.h in main.cpp (like: #include "mydll.h").
- 3. Right-click TestDLL's project and go to Properties.
- 4. Select Linker and Command Line. Type ./MyDLL.lib in Additional Options.

TestDLL Property Pages		X
Configuration: Active(Debug)	Platform: Active(Win32)     Configuration Manager	
General Debugging C/C++ C/C++ Ceneral Input Debugging System Optimization Embedded IDL Advanced	All Options. /OUT:"Debug/TestDLL.exe" /INCREMENTAL /NOLOGO /DEBUG /PDB:"Debug/TestDLL. pdb" /SUBSYSTEM:CONSOLE /MACHINE:X86 kernel32.lib user32.lib gdi32.lib winspool. lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp 32.lib	
<ul> <li>Browse Information</li> <li>Build Events</li> <li>Custom Build Step</li> <li>Web Deployment</li> </ul>	Additional Options:	<
l	OK Cancel Apply Help	,

### Now you can add additional functions to it

For example, let's add a method to CMyDLL class which prints a char \*

object and also modify the constructor to accept a char \* object.

```
CMyDLL(char * strToPrint);
void printStr() const;
```

```
};
Also, in MyDLL.cpp let's add the following code:
CMyDLL::CMyDLL(char * strToPrint) {
    __strToPrint = strdup(strToPrint);
}
void CMyDLL::printStr() const {
    _cout << _strToPrint << endl;
}
After compiling the code and copy the necessary files, your main() can look
like this:
```

#include "mydll.h"

#### int main () {

```
CMyDLL *cmd = new CMyDLL("This is my first DLL");
cmd->printStr();
return 0;
```

}

## That's it; you're done creating you first DLL.