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:

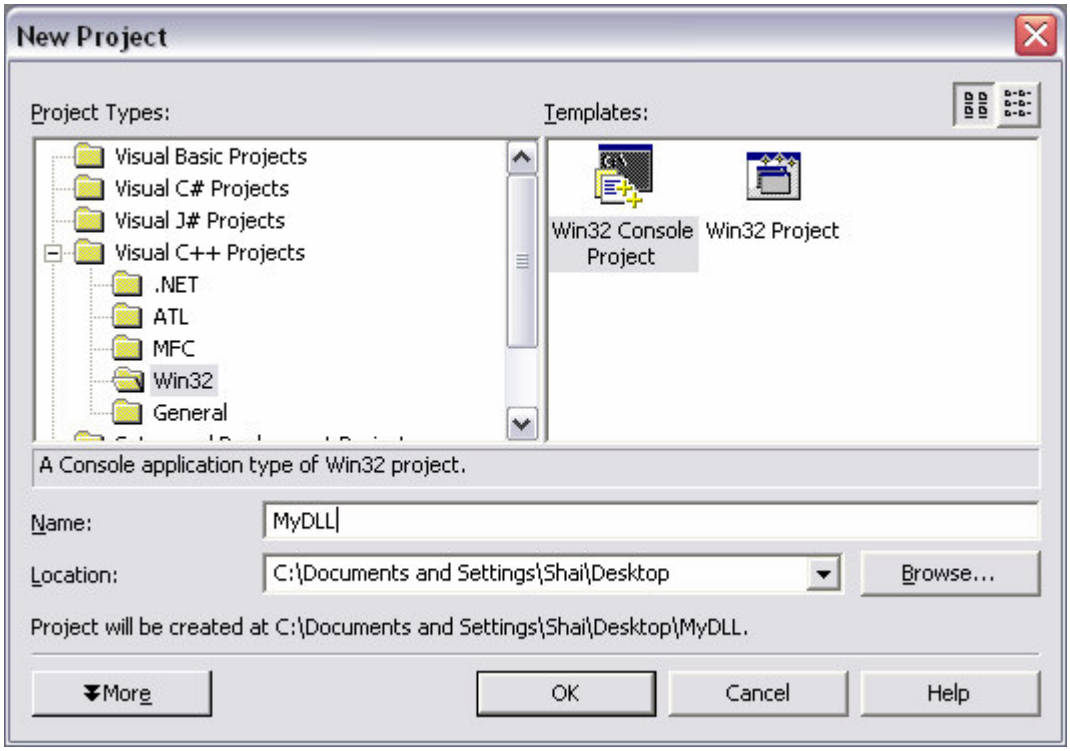
- MyDLL.h – the DLL's header file.
- MyDLL.lib – short static library of the DLL.
- MyDLL.dll – the compiled DLL.

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.

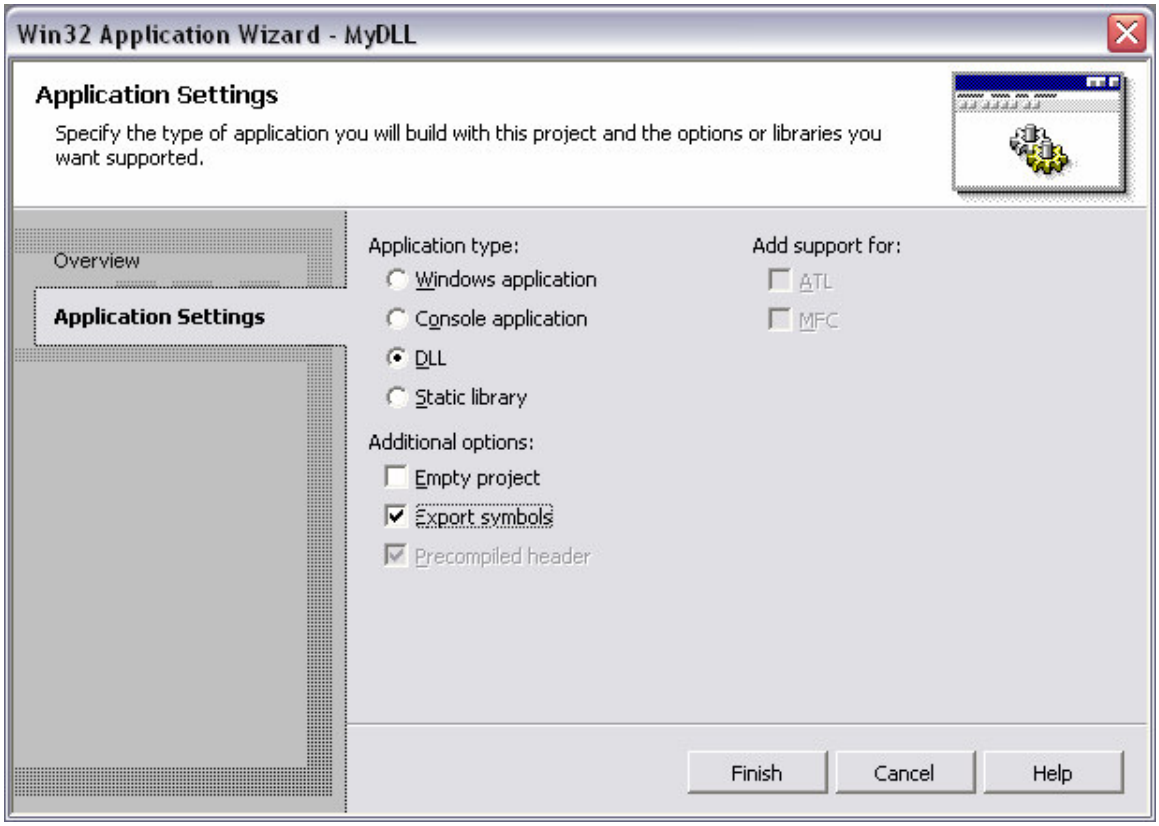
Creating your first DLL

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.



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



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.

MyDll.h

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
```

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 class is exported from the MyDLL.dll
class MYDLL_API CMyDLL {
public:
        CMyDLL(void);
        // TODO: add your methods here.
};
```

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.

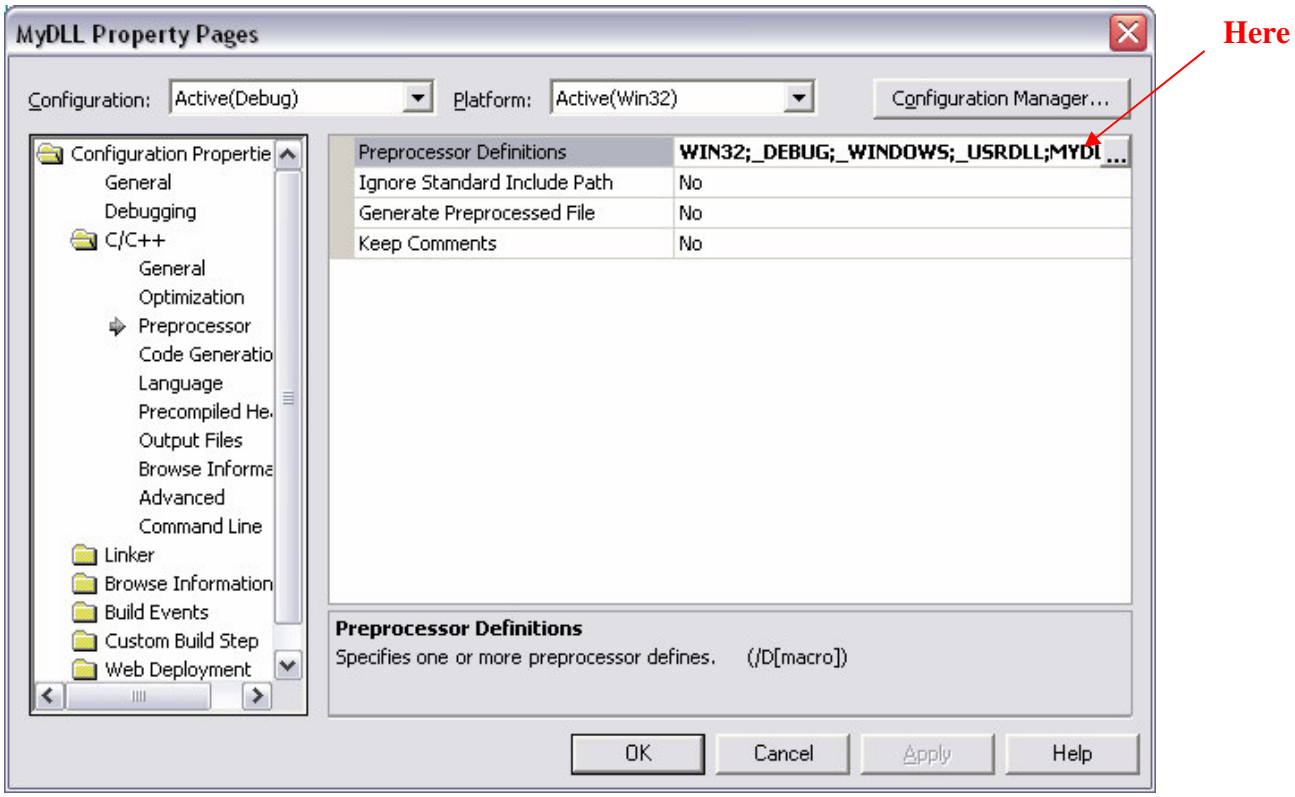
```
extern MYDLL_API int nMyDLL;

MYDLL_API int fnMyDLL(void);
```

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.
  - Go to Configuration Properties – C/C++ and Preprocessor.
  - Make sure that MYDLL_EXPORTS appear in Preprocessor Definitions line.



MyDLL.cpp

This code was added by the wizard.

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

CPP header. Includes, etc.

```
BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved) {
        switch (ul_reason_for_call)    {
                // Called when a process starts.
                case DLL_PROCESS_ATTACH:
                // Called when a thread starts.
                case DLL_THREAD_ATTACH:
                // Called when a thread ends.
                case DLL_THREAD_DETACH:
                // Called when a process ends.
                case DLL_PROCESS_DETACH:
                        break;
        }
   return TRUE;
}
```

The DLL entry point. Look in: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/dllmain.asp 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.

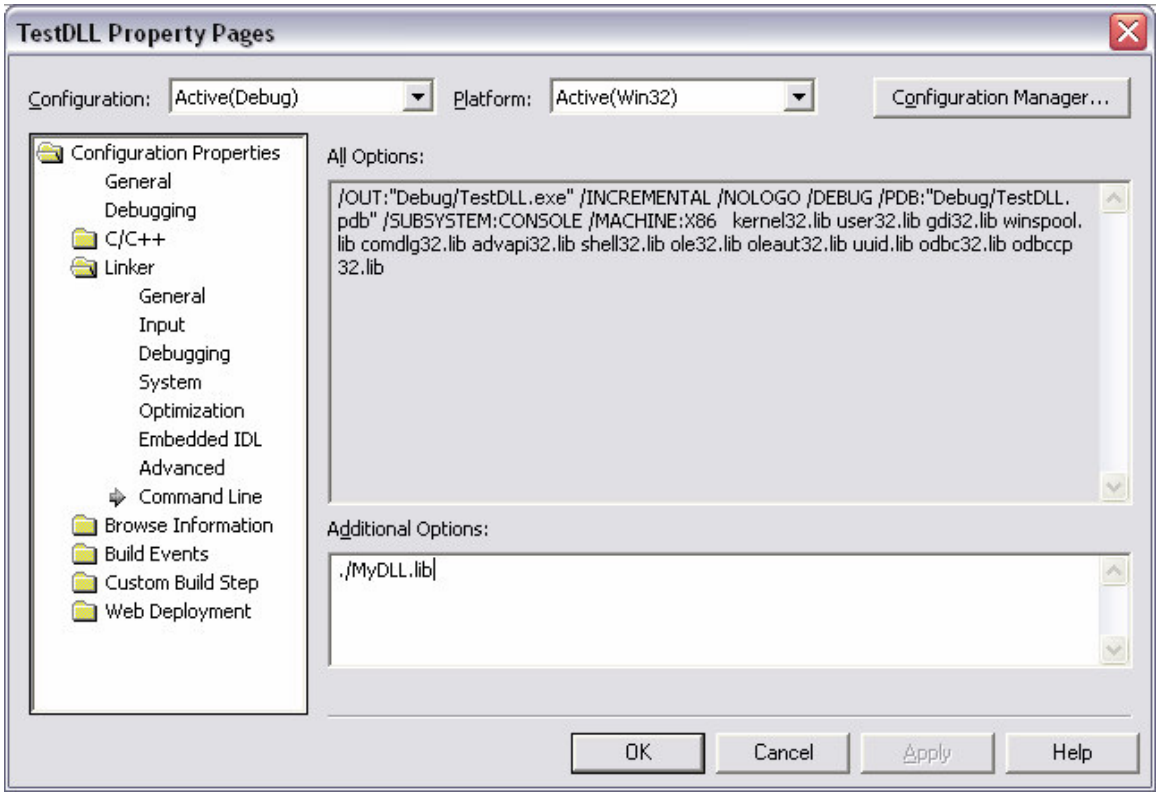
```
// This is the constructor of a class that has been exported.
// see MyDLL.h for the class definition
CMyDLL::CMyDLL() {
        return;
}
```

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.



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.

```
class MYDLL_API CMyDLL {
private:
        char* _strToPrint;

public:
        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.