Creating MySQL UDFs with Microsoft Visual C++ Express

Original Link: http://rpbouman.blogspot.com/2007/09/creating-mysql-udfs-with-microsoft.html

By Ronald Bouman, notes added: sam j levy

Preparation

Before we can actually start, we need to install and configure some software and obtain a few resources.

Installing Visual C++ 2005 Express Edition

sam j levy note: Visual C++ 2010 Express will work, http://www.microsoft.com/express/Downloads/

First, you'll need to download and install Microsoft Visual C++ 2005 Express Edition. At present, this is the latest stable release of the free (As in Beer) version of the popular Visual Studio IDE, set up to create, compile and debug C++ programs.

The installation procedure can take a little while, but is otherwise pretty straightforward.

If you already have a paid-for version of Visual Studio 2005, you should **not** install the express edition (you are at risk of messing up the existing installation if you do). In that case, you should use your paid-for version or alternatively, download the upcoming release (Microsoft Visual C++ 2008 Express Edition, now in Beta) and give that a spin.

Installing the Microsoft Platform SDK

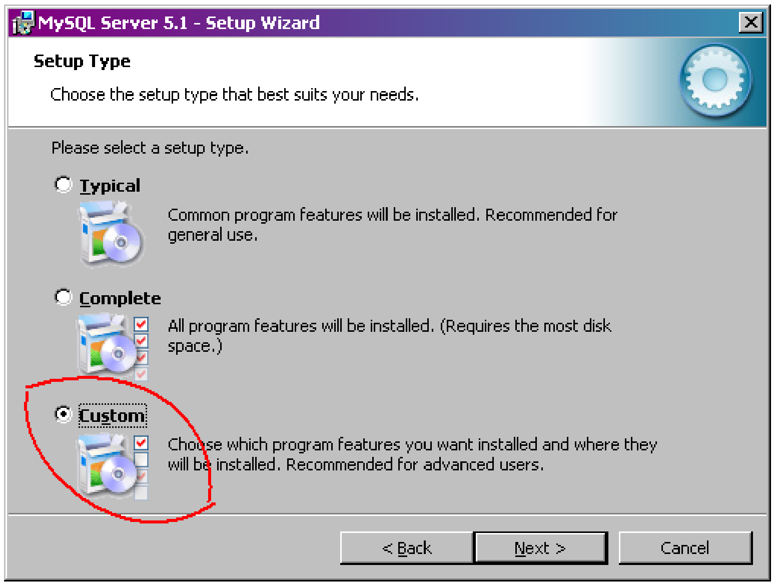
sam j levy note: Windows 7 SDK http://www.microsoft.com/downloads/en/details.aspx?FamilyID=6B6C21D2-2006-4AFA-9702-529FA782D63B

Apart from Visual Studio, you also need to have the Microsoft Platform SDK installed. Although this SDK is officially entitled "Microsoft ® Windows Server® 2003 R2 Platform SDK", it includes the resources for many flavours of Windows, including Windows 2000, Windows 2003 and Windows XP.

Installing the MySQL Development resources

Source files for UDFs contain references to header files supplied by MySQL. The easiest way to obtain them is by installing them using the `Setup.exe` installer program that you use to install the MySQL Server.

If you are installing a new server, you can ensure that the files are installed by choosing "custom" in the "Setup Type" step of the wizard started by the `Setup.exe`:


MySQL Server 5.1 - Setup Wizard

Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

Typical

Common program features will be installed. Recommended for general use.

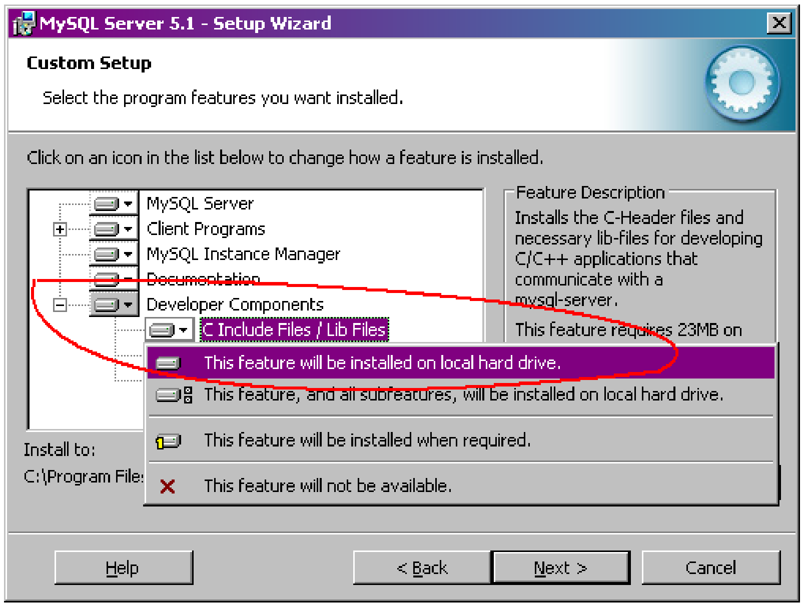
Complete

All program features will be installed. (Requires the most disk space.)

[✓ Custom]

Choose which program features you want installed and where they will be installed. Recommended for advanced users.

After that, you will be able to choose which features you want to install. You need to ensure that the "C Include files/Lib Files" under the "Developer Components" is selected:


MySQL Server 5.1 - Setup Wizard

Custom Setup

Select the program features you want installed.

Click on an icon in the list below to change how a feature is installed.

MySQL Server
Client Programs
MySQL Instance Manager
Documentation
Developer Components
C Include Files / Lib Files

Feature Description

Installs the C-Header files and necessary lib-files for developing C/C++ applications that communicate with a mysql-server.

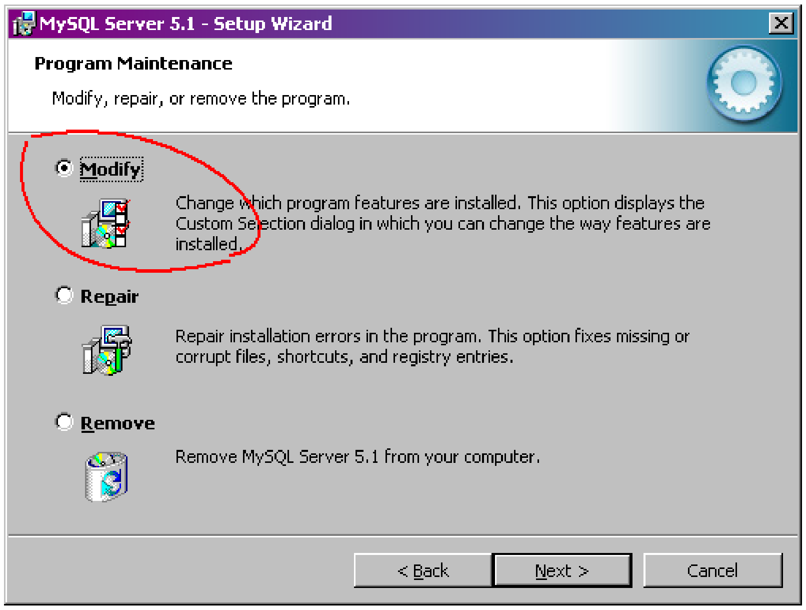
This feature requires 23MB on

This feature will be installed on local hard drive.
This feature, and all subfeatures, will be installed on local hard drive.
This feature will be installed when required.
This feature will not be available.

Install to:
C:\Program File

If you are not installing a new server, you should first check to see if you have an `include` directory immediately beneath the MySQL base direction. If so, you probably don't need to do anything right now.

If you don't have the include directory, it probably means you did not choose to install the "C Include files/Lib Files" when installing the server (by default, they are not installed). Running the `Setup.exe` program again will offer you the possibility to add new components to the installation:


MySQL Server 5.1 - Setup Wizard

Program Maintenance

Modify, repair, or remove the program.

(•) Modify

Change which program features are installed. This option displays the Custom Selection dialog in which you can change the way features are installed.

( ) Repair

Repair installation errors in the program. This option fixes missing or corrupt files, shortcuts, and registry entries.

( ) Remove

Remove MySQL Server 5.1 from your computer.

And from here, you will be led to the step where you can choose to install the include files and library files.

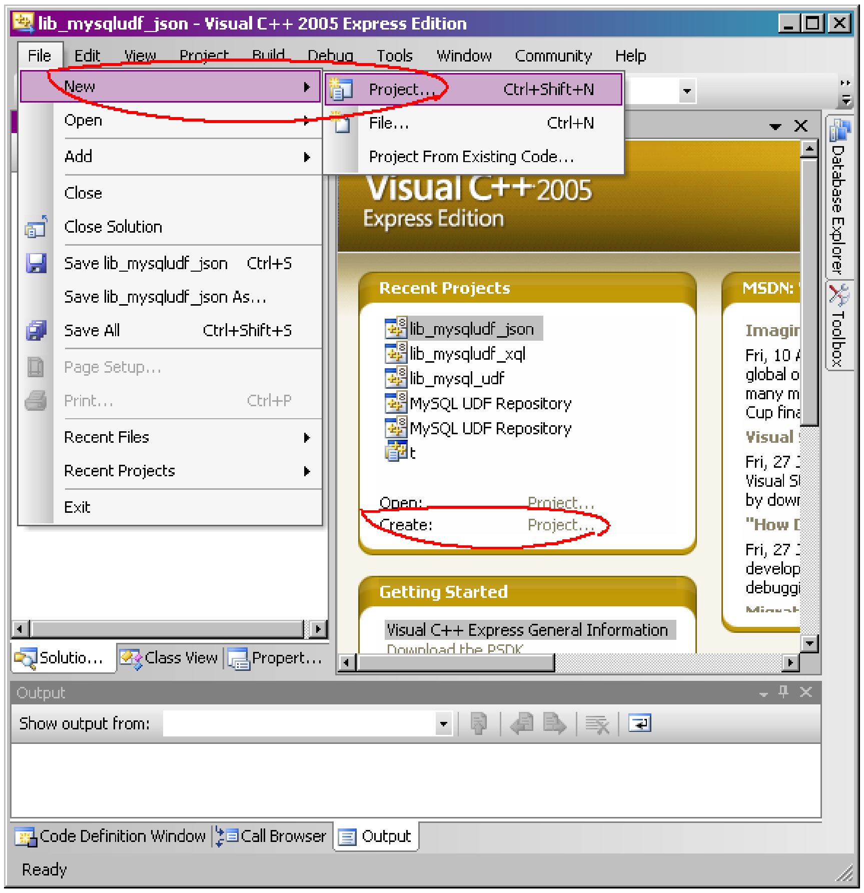
Setting up a VC++EE Project for MySQL UDFs

Once you fulfilled all necessary prerequisites, the next step is to create a Visual Studio Project. In this context, a project is a container for source files, resources, references to existing libraries, as well as a number of options to compile the source files.

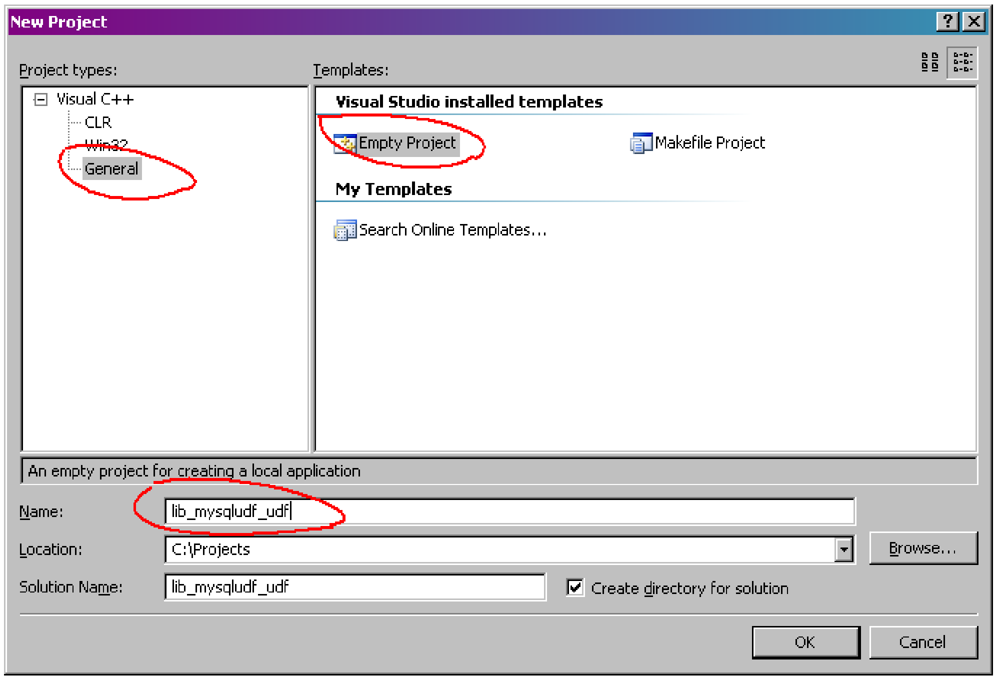
In this article we will set up a VS project to compile the (existing) source of lib_mysqludf_udf. This is a library that demonstrates the basic features of the MySQL UDF interface, such as processing arguments, returning values and handling errors. It is great to get started programming MySQL UDFs. In order to walk through this example, you only need to download the C source file. (Tip: right click the link and choose an option from the context menu to download the file to your local file system.)

Creating a new Project

To actually create the project, we can use the File/New/Project... menu or else the "create project" hyperlink on the the Visual Studio Startpage:



This opens a dialog where we must enter a few details about our project:



An empty project for creating a local application

Name: lib_mysqludf_udf

Location: C:\Projects

Solution Name: lib_mysqludf_udf

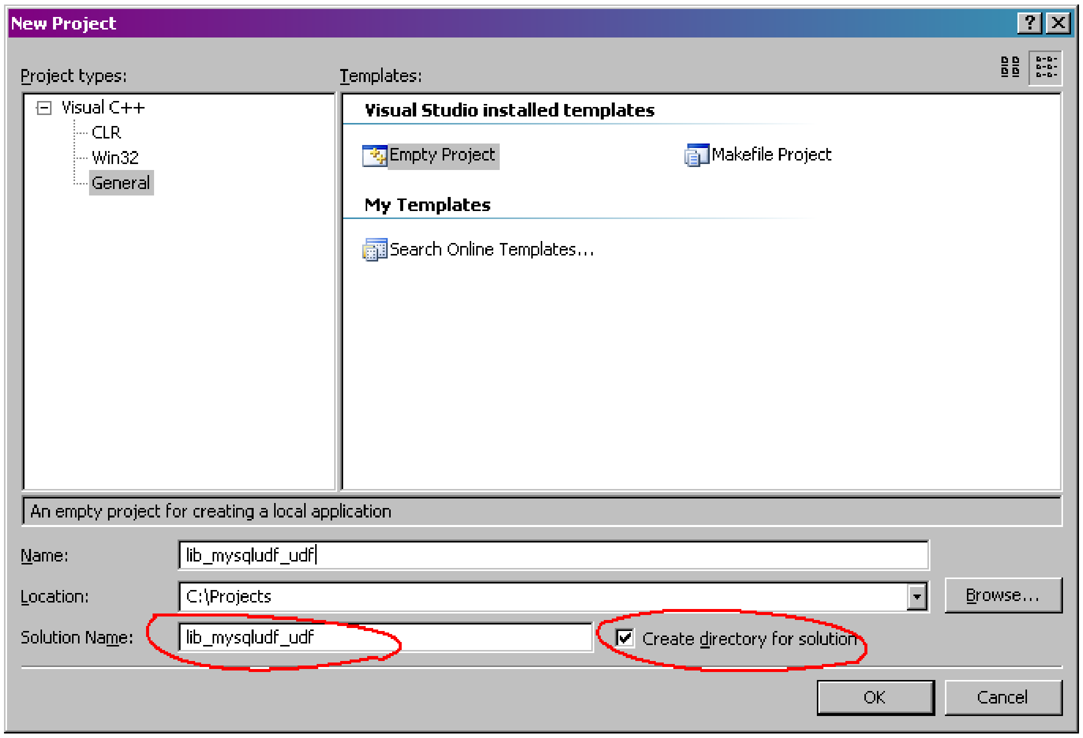
[✓] Create directory for solution

For the Visual C++ Express edition, it works best to choose a General/Empty Project. (The paid-for edition of Visual Studio provides templates for projects to create dynamically linked libraries a.k.a. DLLs but as we shall see later on we have to configure this manually.)

We are also required to provide a name for the project. In this case, we use a name that corresponds directly to the source file: `lib_mysqludf_udf`.

Visual Studio Solution

In Visual Studio, a project is always part of a Solution, which is basically a container for a number of related projects. Because we just started a new project, we are implicitly creating a new solution too, so we have to specify a few things about that as well:



An empty project for creating a local application

Name: lib_mysqludf_udf

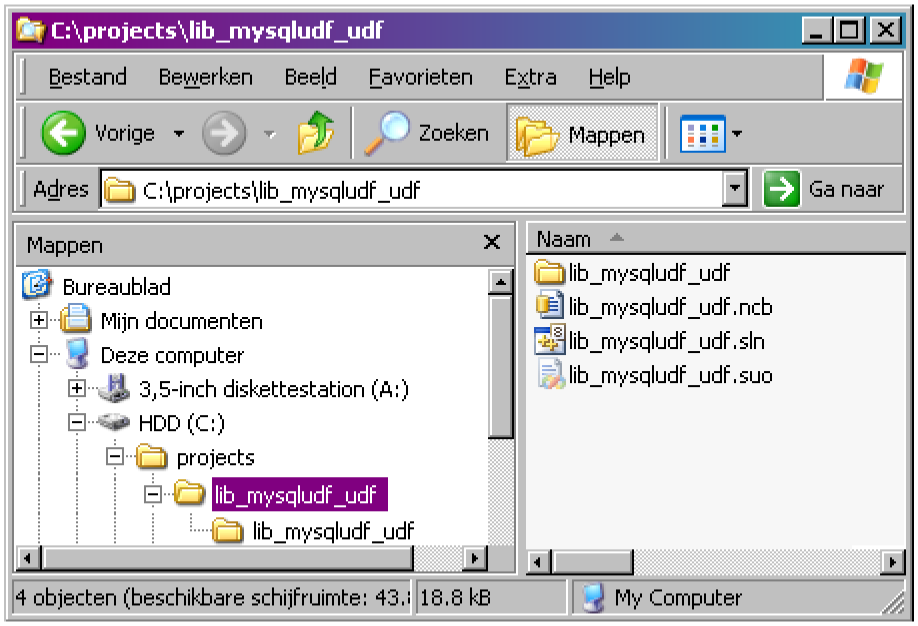
Location: C:\Projects

Solution Name: lib_mysqludf_udf

[✓] Create directory for solution

In this case, we create a separate directory for the solution itself, and we use the same name for the solution as for the project. It is important to realize that there can be multiple projects per solution, in which case it probably makes more sense to choose a distinct name for the solution as a whole.

After confirming the dialog, a number of directories and files are created:



C:\projects\lib_mysqludf_udf

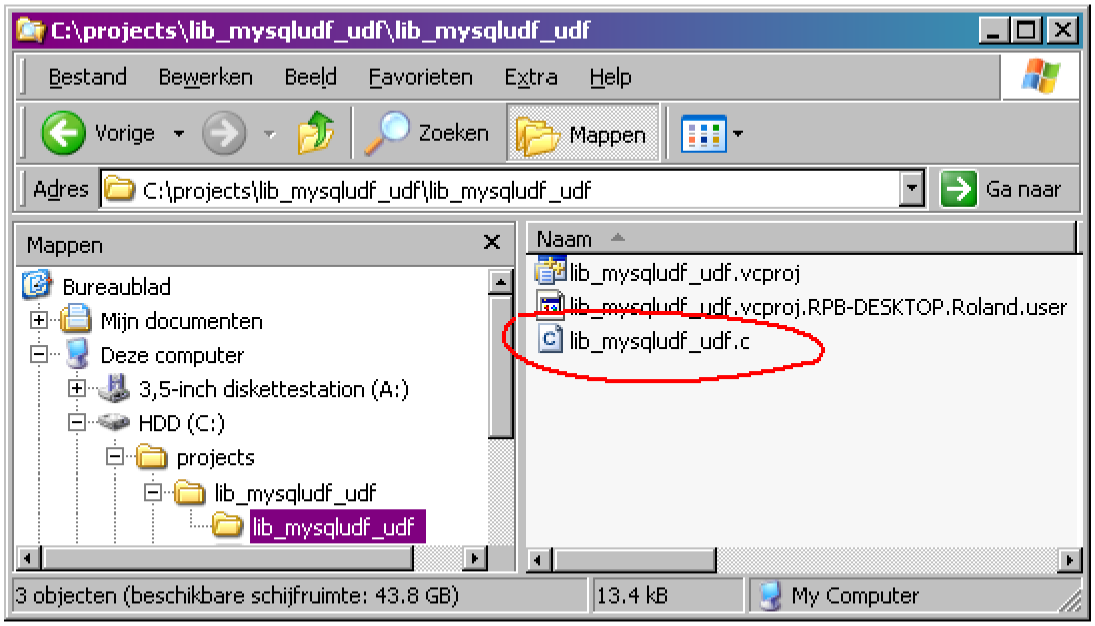
Bureaublad
Mijn documenten
Deze computer
3,5-inch diskettestation (A:)
HDD (C:)
projects
lib_mysqludf_udf
lib_mysqludf_udf

lib_mysqludf_udf
lib_mysqludf_udf.ncb
lib_mysqludf_udf.sln
lib_mysqludf_udf.suo

4 objecten (beschikbare schijfruimte: 43.: 18.8 kB My Computer

Adding a source file

Now it is time to add the source file to our project, so if you didn't download the `lib_mysqludf_udf` C source file yet, you should do so now. Be sure to copy the `lib_mysqludf_udf.c` source file to the `lib_mysqludf_udf` project directory beneath the `lib_mysqludf_udf` solution directory:



C:\projects\lib_mysqludf_udf\lib_mysqludf_udf

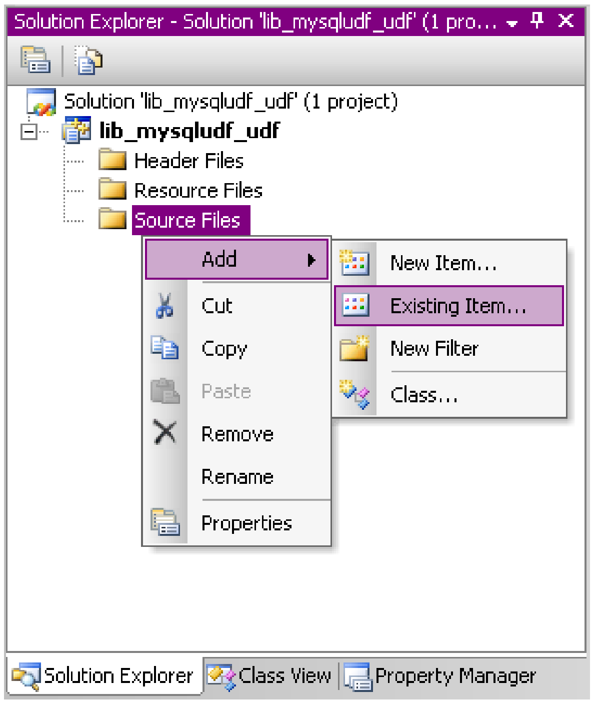
Bureaublad
Mijn documenten
Deze computer
3,5-inch diskettestation (A:)
HDD (C:)
projects
lib_mysqludf_udf
lib_mysqludf_udf

lib_mysqludf_udf.vcproj
lib_mysqludf_udf.vcproj.RPB-DESKTOP.Roland.user
lib_mysqludf_udf.c

3 objecten (beschikbare schijfruimte: 43.8 GB) 13.4 kB My Computer

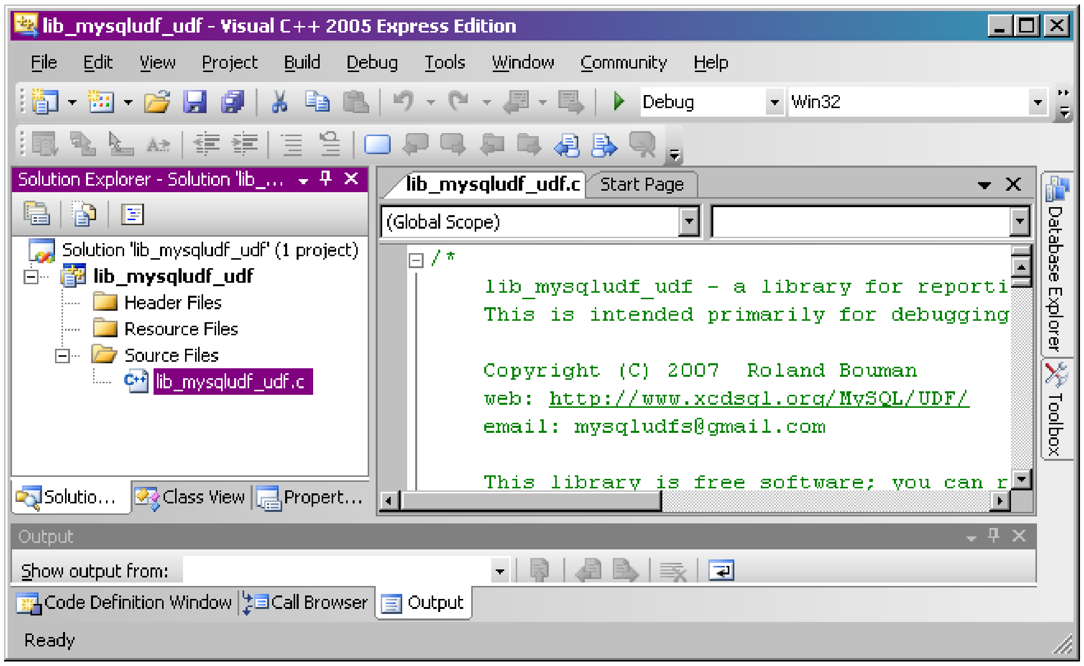
Copying the source file there is just a matter of convenience - I like to keep things that belong together in one place. If you don't keep the file in the project directory, things might may (and probably will) go wrong if you move the source file or the project to another location later on.

Copying the file to the directory still does not formally add the file to the project. To actually add the file to the project, you can right-click the "Source Files" folder beneath the project folder in the Visual Studio Solution Explorer window and use the context menu to add the existing item:



As an alternative to using the menu you can also add the source file to your project by simply dragging it into the Solution Explorer and dropping it into the "Source Files" folder of the project.

If all went well, the source file is now part of the project and can be opened from the solution explorer:



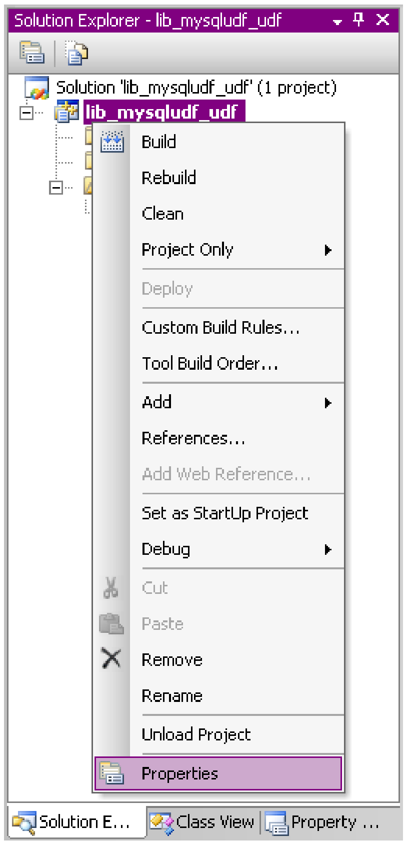
```
/*
    lib_mysqludf_udf - a library for reporti
    This is intended primarily for debugging

    Copyright (C) 2007  Roland Bouman
    web: http://www.xcdsql.org/MySQL/UDF/
    email: mysqludfs@gmail.com

    This library is free software; you can r
```

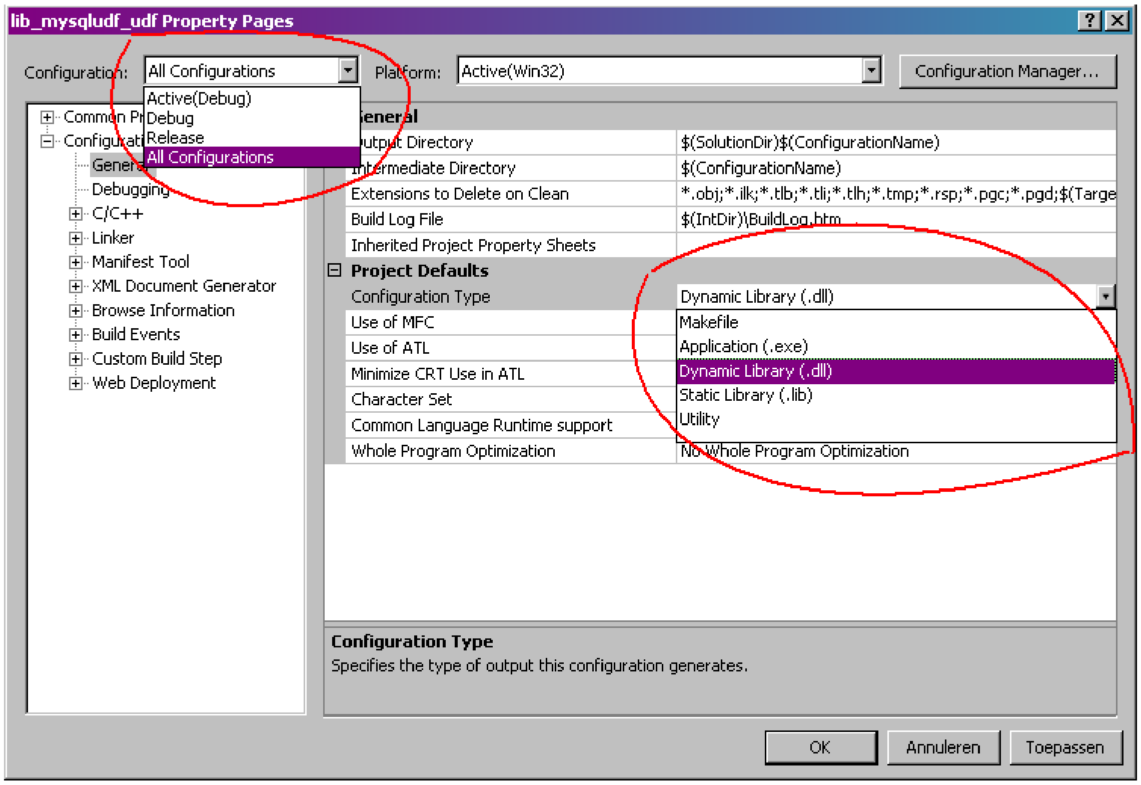
Project Configuration

Although we already defined the structure of the project, we need to configure it in order to compile it. The configuration can be edited through a single dialog which can be accessed by clicking the "Properties" item in the project folder's context menu:



General Properties

We first need to take care of the general configuration. A Visual Studio project can have several configurations - something which is very useful if you want to create different builds (debug or release) from the same project. However, it is a good idea to first configure all project properties that are the same for all configurations. To do that, we choose "All configurations" in the top left listbox of the configuration dialog. (For this example, we do not separately configure for debug and release builds.)


lib_mysqludf_udf Property Pages

Configuration: All Configurations (Active(Debug), Debug, Release, All Configurations)
Platform: Active(Win32)

General	
Output Directory	$(SolutionDir)$(ConfigurationName)
Intermediate Directory	$(ConfigurationName)
Extensions to Delete on Clean	*.obj;*.ilk;*.tlb;*.tli;*.tlh;*.tmp;*.rsp;*.pgc;*.pgd;$(Targe
Build Log File	$(IntDir)\BuildLog.htm
Inherited Project Property Sheets	
Project Defaults	
Configuration Type	Dynamic Library (.dll)
Use of MFC	
Use of ATL	
Minimize CRT Use in ATL	
Character Set	
Common Language Runtime support	
Whole Program Optimization	No Whole Program Optimization

Configuration Type options: Makefile, Application (.exe), Dynamic Library (.dll), Static Library (.lib), Utility

Configuration Type
Specifies the type of output this configuration generates.

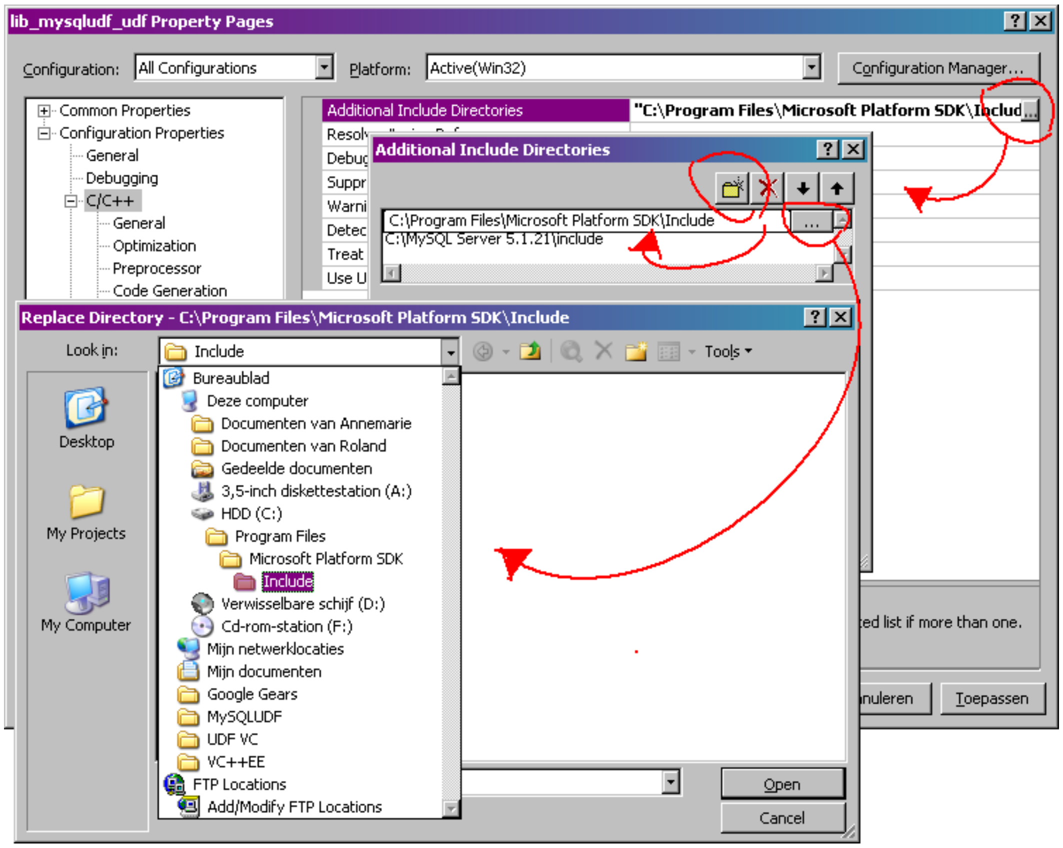
The rest of the configuration process is a matter of editing individual properties. Related properties are organized in property pages, each of which covers a particular aspect of the project. By default, the "General" property page is selected and it makes sense to start editing properties there right away.

In the "General" property page we need to set the "Configuration Type" property to "Dynamic library (.dll)" as we need to be able to load the UDF library dynamically into the MySQL Server.

Configuring the Include path

MySQL UDFs refer to types and contants defined in C header files provided by MySQL. In turn, these refer to header files from the Microsoft Platform SDK. The project does not know automatically where to locate these header files, so we need to configure the project and point it to the location(s) manually.

To specify the location of the header files, we need to activate the "C/C++" property page and edit the "Additional Include Directories" property. You can either directly type the paths in the text box, or otherwise click the elipsis buttons (...) to browse for them.



Configuration: All Configurations
Platform: Active(Win32)
Additional Include Directories: "C:\Program Files\Microsoft Platform SDK\Includ...

C:\Program Files\Microsoft Platform SDK\Include
C:\MySQL Server 5.1.21\include

For this example, we need to specify two locations:

- The location of the "include" directory beneath the MySQL installation directory.
  - sam j levy note: C:\Program Files\MySQL\MySQL Server 5.5\include
- The location of the "include" directory beneath the Microsoft Platform SDK installation directory.
  - sam j levy note: C:\Program Files\Microsoft SDKs\Windows\v7.0A\Include

If you can't find these directorie, you most likely need to revisit the "preparation" section of this article.

Adding the `HAVE_DLOPEN` macro

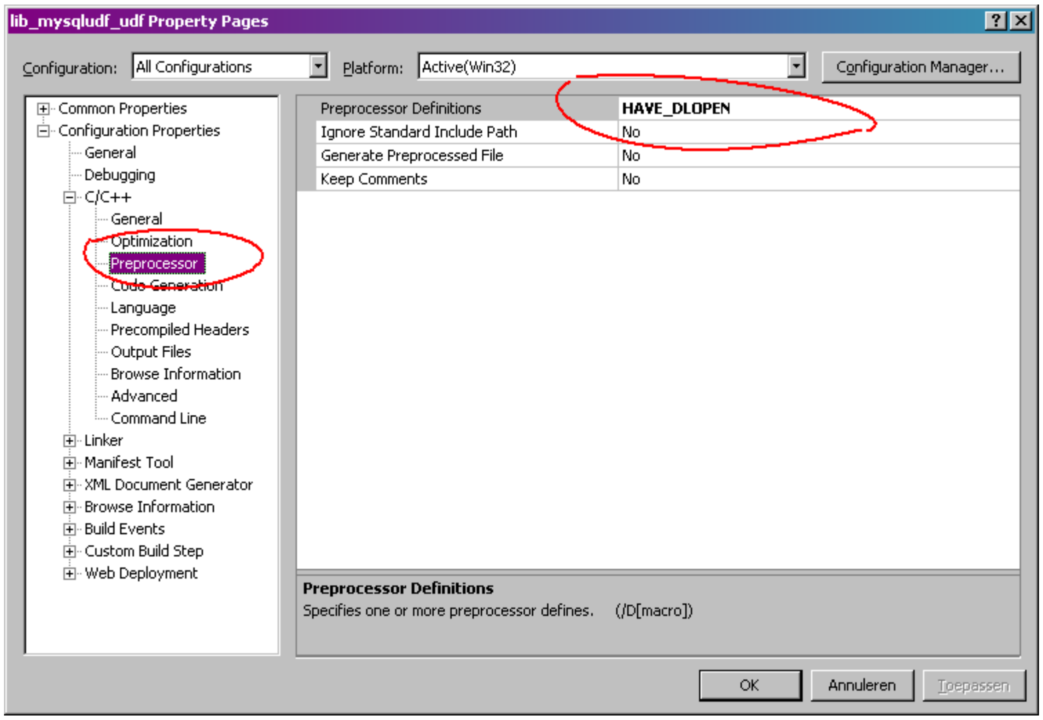
The `lib_mysqludf_udf.c` source file was created using the `udf_example.c` source file from the MySQL source distribution as an example. The structure of that code uses conditional compilation according to wheter `HAVE_DLOPEN` is defined:

```
#ifdef HAVE_DLOPEN

...code goes here...

#endif /* HAVE_DLOPEN */
```
And this is also used in `lib_mysqludf_udf`.

I admit that I don't understand why that is there, or what it is supposed to achieve, and I would very much like someone to comment on this blog entry to explain it. Anyway, for Visual C++ it means we have to explicitly define it using a Preprocessor definition:


lib_mysqludf_udf Property Pages

Configuration: All Configurations   Platform: Active(Win32)   Configuration Manager...

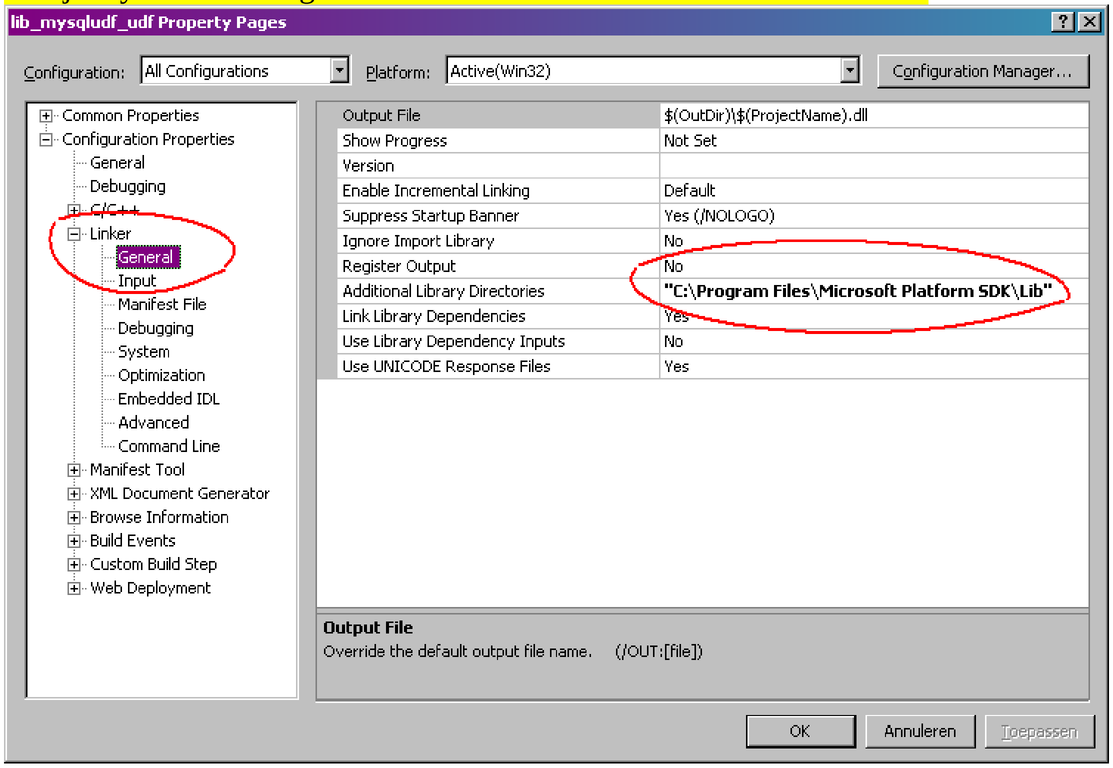
Preprocessor Definitions	HAVE_DLOPEN
Ignore Standard Include Path	No
Generate Preprocessed File	No
Keep Comments	No

Preprocessor Definitions
Specifies one or more preprocessor defines. (/D[macro])

Configuring the library path

We configured the project to compile a Dynamic-Link Library. For the compiler, this means it cannot just compile the code and package it in a file: the dll target file needs to adhere to a certain specification. In order to make that happen, it needs to link to existing libraries from the platform SDK.

Just like we did for the include path, we need to tell Visual C++ where it can find the libraries it must link to. This can be configured by editing the "Additional Library Directories" property in the "Linker" property page: sam j levy note: C:\Program Files\Microsoft SDKs\Windows\v7.0A\Lib


lib_mysqludf_udf Property Pages

Configuration: All Configurations   Platform: Active(Win32)   Configuration Manager...

Output File	$(OutDir)\$(ProjectName).dll
Show Progress	Not Set
Version	
Enable Incremental Linking	Default
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Additional Library Directories	"C:\Program Files\Microsoft Platform SDK\Lib"
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Use UNICODE Response Files	Yes

Output File
Override the default output file name. (/OUT:[file])

In this case, we only need to specify the path of the "Lib" directory find immediately beneath the Platform SDK installation directory.

sam j levy note: **Adding The Module Definition File**

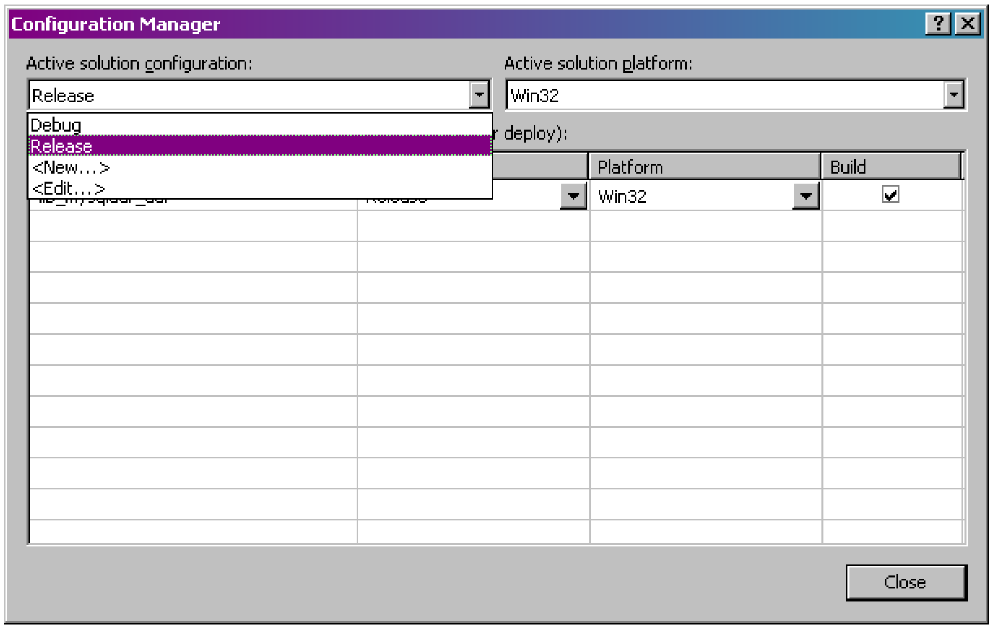
- 'Add Existing' the .def file associated with the UDF source code
- Under Configuration Properties/Linker/Input/Module Definition File type in the file name of the .def you just added to the project, example: myudf.def

If your UDF Source doesn't come with a module definition file, make one, <udf_name>.def
The format looks like:

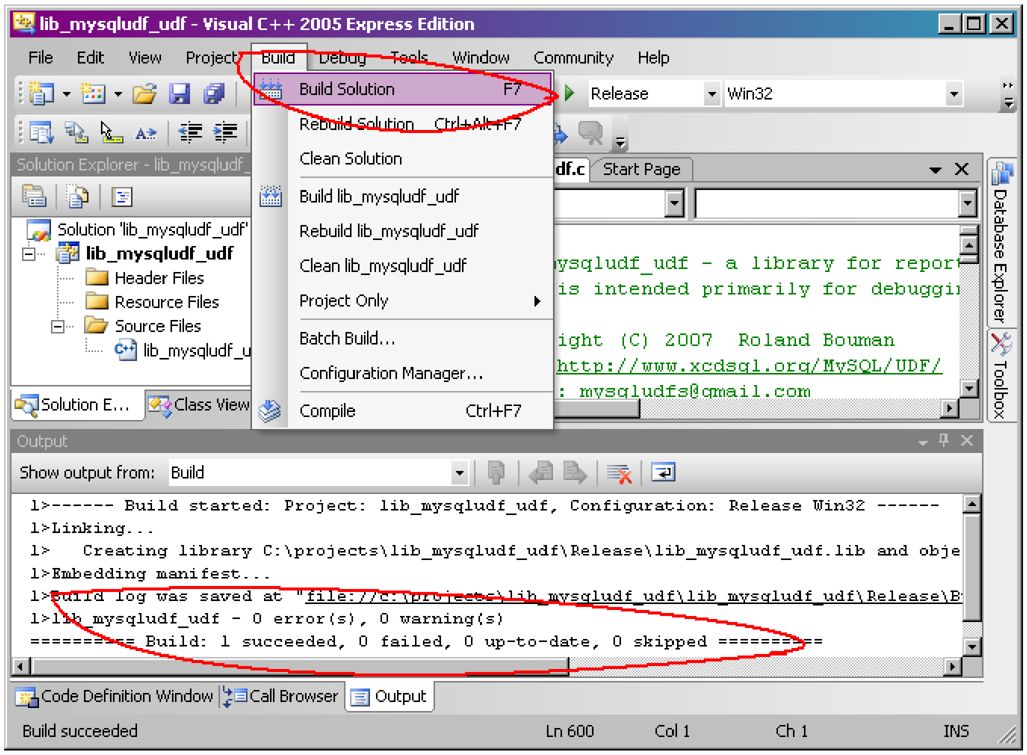
```
LIBRARY <udf_name>
EXPORTS
     <list each function, example:>
     damlev_init
     damlev_deinit
     damlev
```

Compiling the UDFs

At this point, we are ready to compile the project and/or solution. In most cases, you will want to choose the build configuration to choose between a debug or a release build. This can be done by clicking the "Configuration Manager" item in the build menu to invoke the Configuration Manager dialog:



Actually building the project is done using the "Build Solution" or " Build Project" item in the "Build" menu:



```
ysqludf_udf - a library for report
is intended primarily for debuggi
ight (C) 2007  Roland Bouman
http://www.xcdsql.org/MySQL/UDF/
: mysqludfs@gmail.com
```

```
1>------ Build started: Project: lib_mysqludf_udf, Configuration: Release Win32 ------
1>Linking...
1>   Creating library C:\projects\lib_mysqludf_udf\Release\lib_mysqludf_udf.lib and obje
1>Embedding manifest...
1>Build log was saved at "file://c:\projects\lib_mysqludf_udf\lib_mysqludf_udf\Release\B
1>lib_mysqludf_udf - 0 error(s), 0 warning(s)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

The result of building the solution should be as indicated in the screenshot. If the final line does not read

```
1>lib_mysqludf_udf - 0 error(s), 0 warning(s)
```

you might want to read the remainder of this section to figure out what the problem is.

Common problems

Of course, no programming task is complete without running into trouble. In this section, a few common problems compiling the project are listed, as well as their solutions.

fatal error C1083: Cannot open include file: '*filename.h*'

The output of the build process might look something like this:

```
1>Compiling...
1>lib_mysqludf_udf.c
1>..\..\..\temp\lib_mysqludf_udf.c(41) : fatal error C1083: Cannot open include file:
'my_global.h': No such file or directory
1>Build log was saved at
"file://c:\projects\lib_mysqludf_udf\lib_mysqludf_udf\Release\BuildLog.htm"
1>lib_mysqludf_udf - 1 error(s), 0 warning(s)
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

This indicates that you did not configure the include path properly. You should revisit the section on configuring the include path and ensure that the path does in fact point to the include directory that appears under the mysql installation directory.

The build output might look something like this:

```
1>Compiling...
1>lib_mysqludf_udf.c
1>C:\MySQL Server 5.1.21\include\config-win.h(30) : fatal error C1083: Cannot open
include file: 'winsock2.h': No such file or directory
1>Build log was saved at
"file://c:\projects\lib_mysqludf_udf\lib_mysqludf_udf\Release\BuildLog.htm"
1>lib_mysqludf_udf - 1 error(s), 0 warning(s)
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

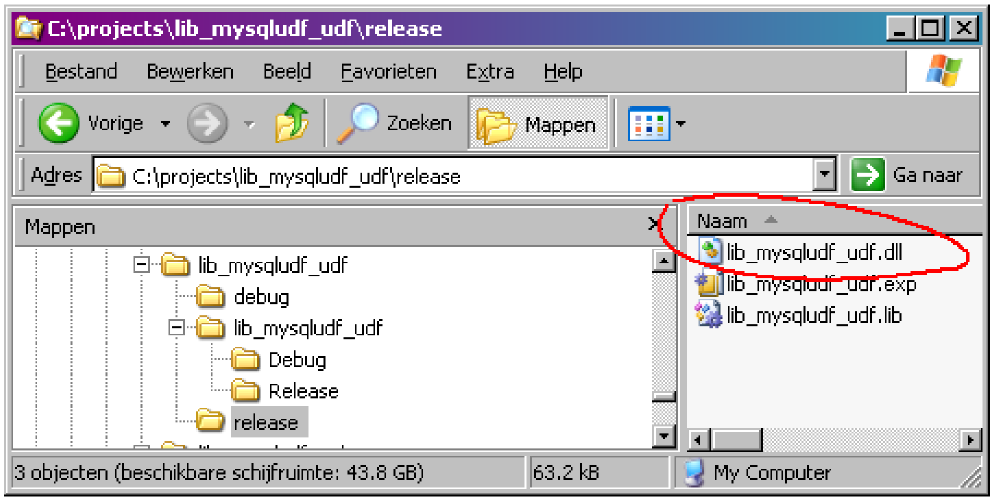
This is a similar problem. It occurs when you did include the "include" directory beneath the MySQL installation directory but forgot the one beneath the Microsoft Platform SDK installation directory. Because the latter is referenced by the former, both have to be added to the include path.

fatal error LNK1104: cannot open file '*file.lib*'

Your build output might look like this:
```
1>------ Build started: Project: lib_mysqludf_udf, Configuration: Release Win32 ------
1>Linking...
1>LINK : fatal error LNK1104: cannot open file 'uuid.lib'
1>Build log was saved at
"file://c:\projects\lib_mysqludf_udf\lib_mysqludf_udf\Release\BuildLog.htm"
1>lib_mysqludf_udf - 1 error(s), 0 warning(s)
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```
This indicates that you did not properly configure the path where Visual Studio looks for Additional Libraries. You should revisit the relevant section in this article and ensure the configured path does in fact contain the specified missing library.

Installing the UDFs

If you managed to succesfully compile the project you are ready to install the UDFs in your MySQL Server. Depending on the whether you chose to do a "Release" or a "Debug" build, you will find the `lib_mysqludf_udf.dll` in the "Release" or "Debug" directory directly beneath the Solution directory respectively:



The .ddl needs to be copied to a location that is accessible to the MySQL Server. For MySQL versions lower than 5.1.19, the `bin` and/or `lib` directories right beneath the MySQL installation directory should work. For MySQL version 5.1.19 and beyond, you are required to copy the dll to the `plugin_dir`. The `plugin_dir` can be determined by running the following query:
```
mysql> show variables like 'plugin_dir';
+---------------+----------------------------+
| Variable_name | Value                      |
+---------------+----------------------------+
| plugin_dir    | C:\MySQL Server 5.1.21\lib/ |
+---------------+----------------------------+
1 row in set (0.00 sec)
```

When the dll is in place, the UDFs can be installed using the `CREATE FUNCTION` syntax:

```
CREATE FUNCTION lib_mysqludf_udf_info
RETURNS STRING
SONAME 'lib_mysqludf_udf.dll';
```

Common Problems

Even if you successfully compiled the solution, you still might run into a few problems at this stage. A few common ones are described in the remainder of this section.

ERROR 1126 (HY000): Can't open shared library '*file.dll*'

If you encounter this error, it means that MySQL cannot find the library you are referring to in the `SONAME` clause of the `CREATE FUNCTION` statement. You may have made a typo in your statement, or the MySQL may be looking in another location for the libarary than you might think it does. Verify that you typed the correct location. For MySQL 5.1.18 and earlier, ensure that the dll is copied to either the bin and/or lib directory beneath the MySQL installation directory. For MySQL 5.1.19 and beyond, ensure that the file is located in the `plugin_dir`.

ERROR 1127 (HY000): Can't find symbol '*functionname*' in library

If you encounter this error, a few things might be the matter. You might have made a typo in the function identifier in the `CREATE FUNCTION` statement. Another possibility is that you forgot to add the `HAVE_DLOPEN` macro to the preprocessor definitions. If needed, revisit that section in this article.
sam j levy note: Or because you didn't add a Module Definition File, see section 'Adding a Module Definition File'

ERROR 1046 (3D000): No database selected

This error occurs when you did not set the default database. The workaround is to set any database as default database using the `USE` statement:

```
USE test;
```

. Arguably this is a bug in the MySQL Server: it somehow thinks we are trying to create a stored function which is bound to a database. It's as if MySQL cannot distinguish between a UDF and a stored function at this point.

ERROR 1044 (42000): Access denied for user '*user*'@'*host*' to database 'mysql'

This error occurs when the user that is trying to create the function is not privileged to write to the `mysql` system database. To the best of my knowledge, UDFs are written only the `mysql.func` table so I would expect that granting privileges on that table would be enough to be allowed to create UDFs. It turns out that this is not the case. Granting all privileges on the mysql database does allow a user to install UDFs, but I don't know if that is indeed the minimal set of privileges required to install UDFs.