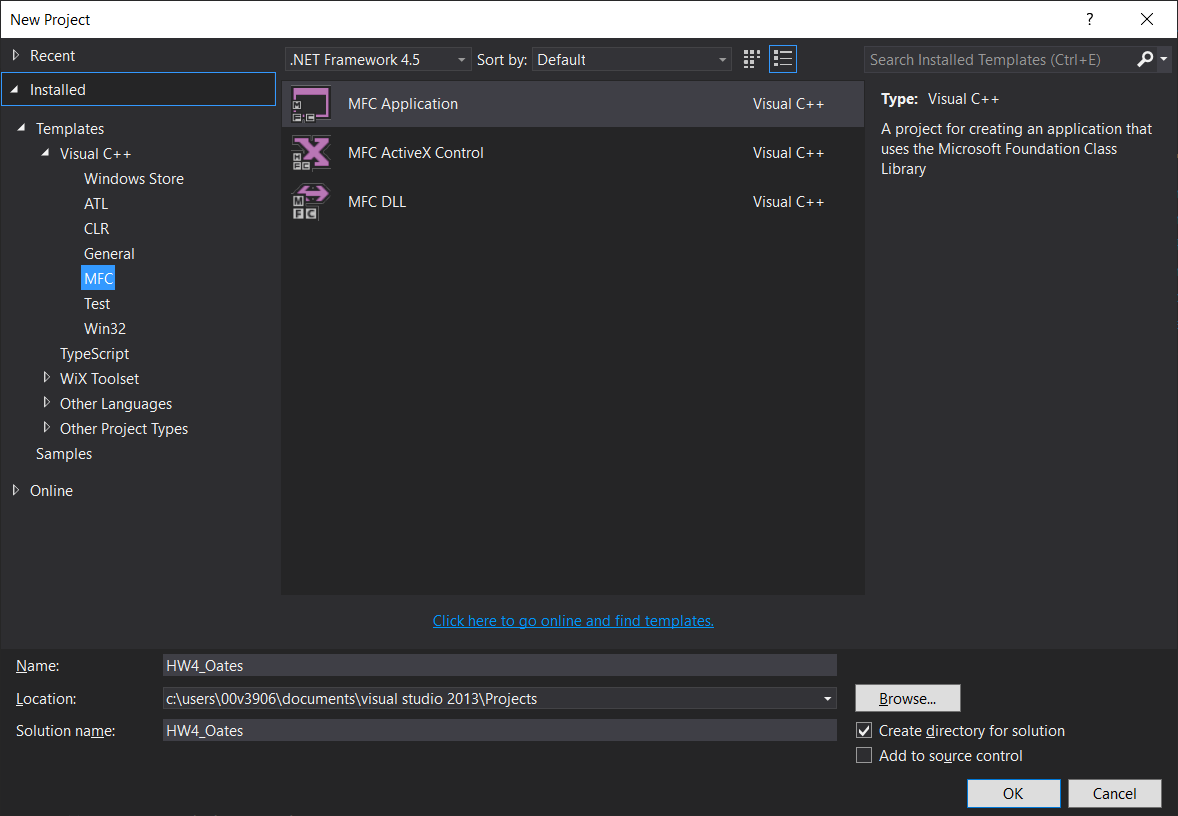
Step by Step Setup Guide

For an MFC Project

1. Launch Visual Studio and go to File > New > Project…

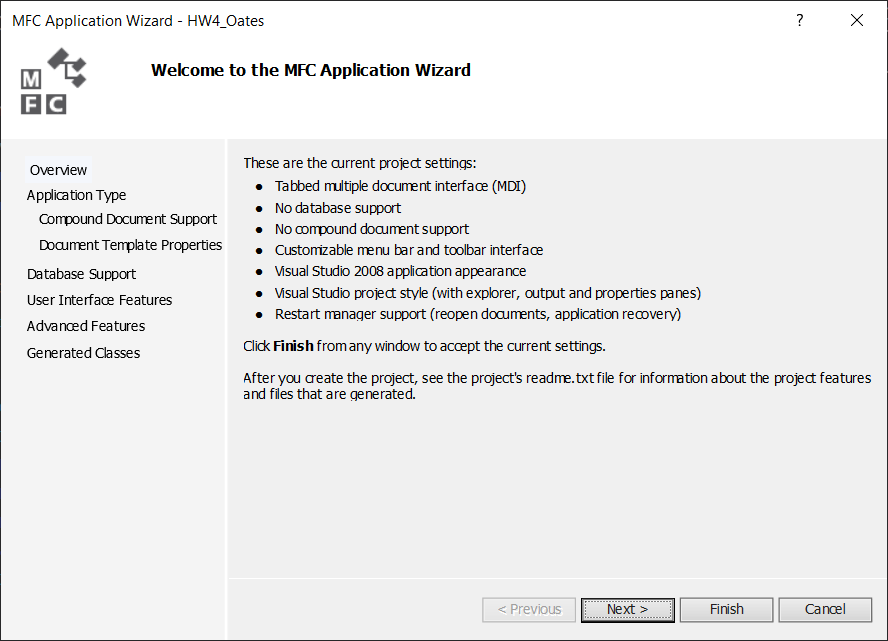
Select MFC project and give it a name (HW4\_<your name>)

If you do not see the option to create an MFC project, try looking in the “Online” group of projects (bottom of the left pane below). If you are still having trouble finding it, let’s talk. I’m sure it is still an option.

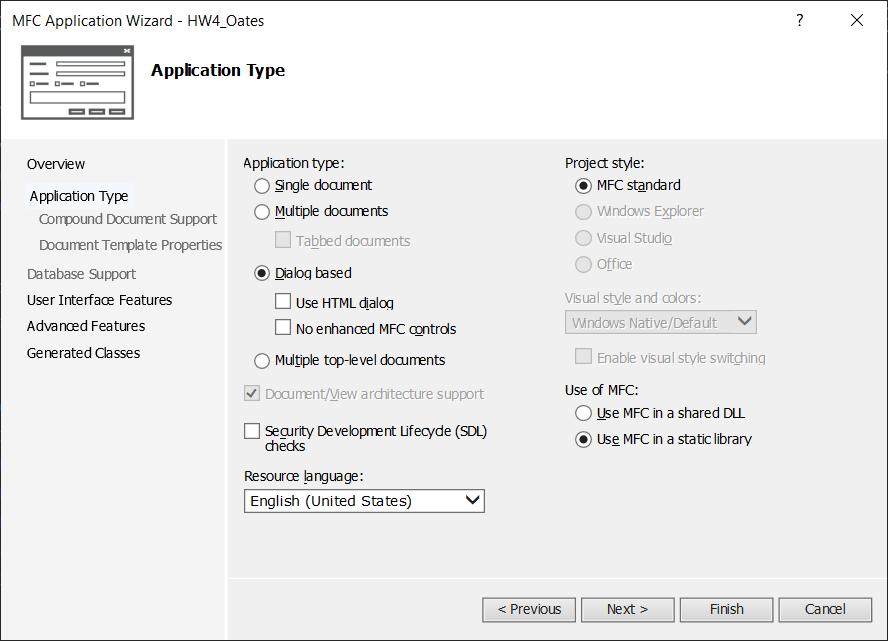


Press OK

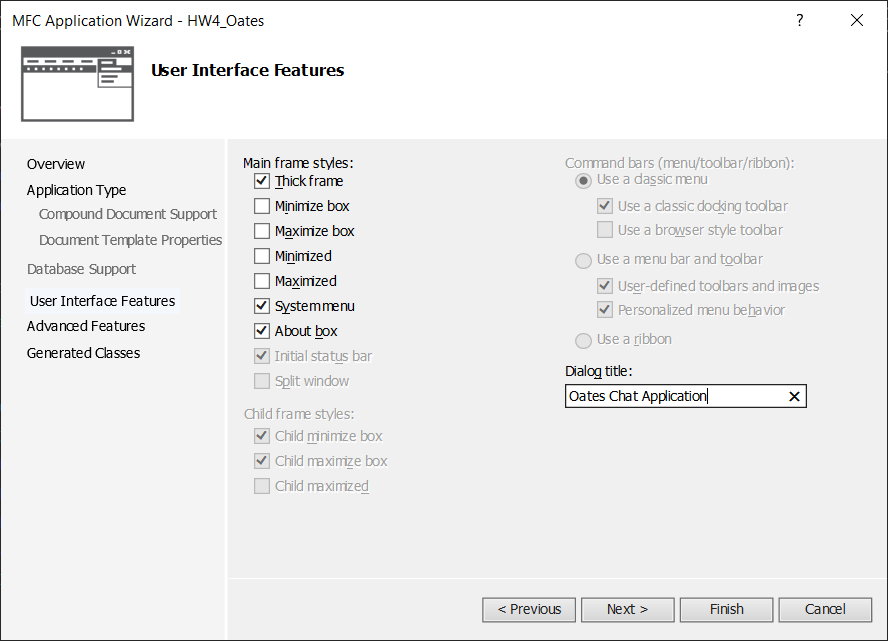
1. The MFC Application Wizard will appear next. Press Next to begin selecting your application settings.



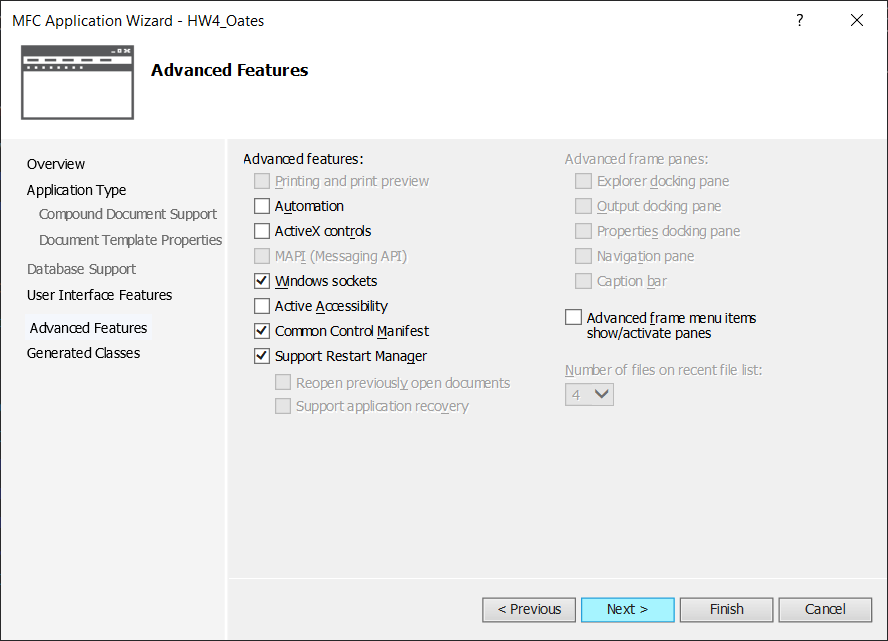
1. Match my selections in the screen shot below and press Next.



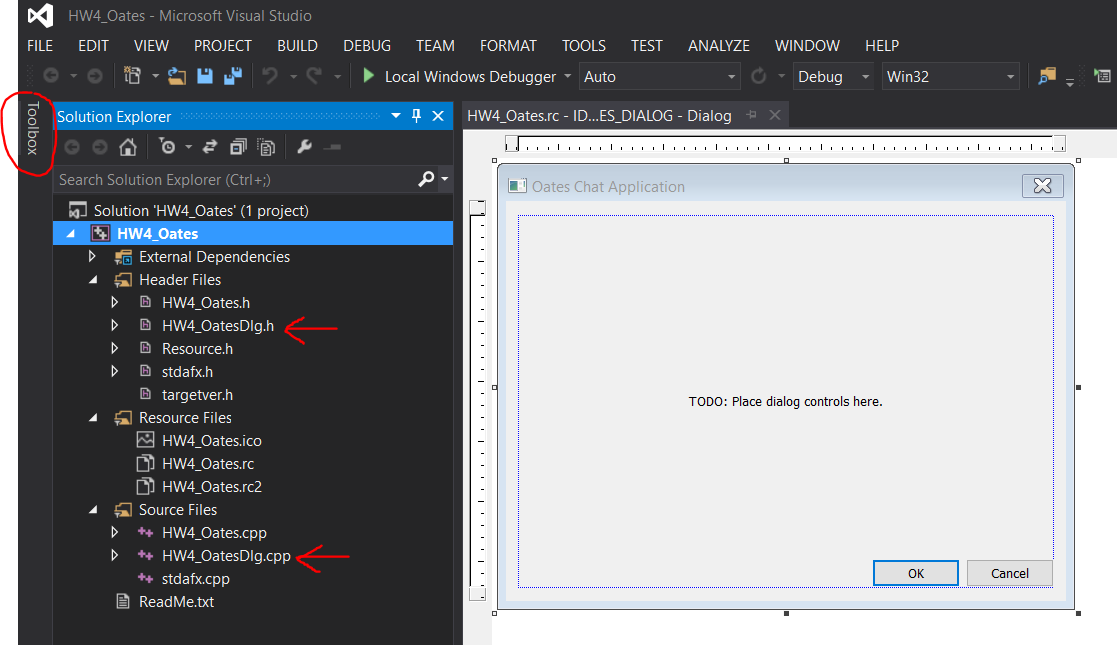
1. The next window lets you set user interface features ( below). Match my settings and press Next.

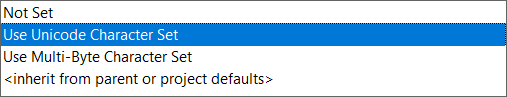
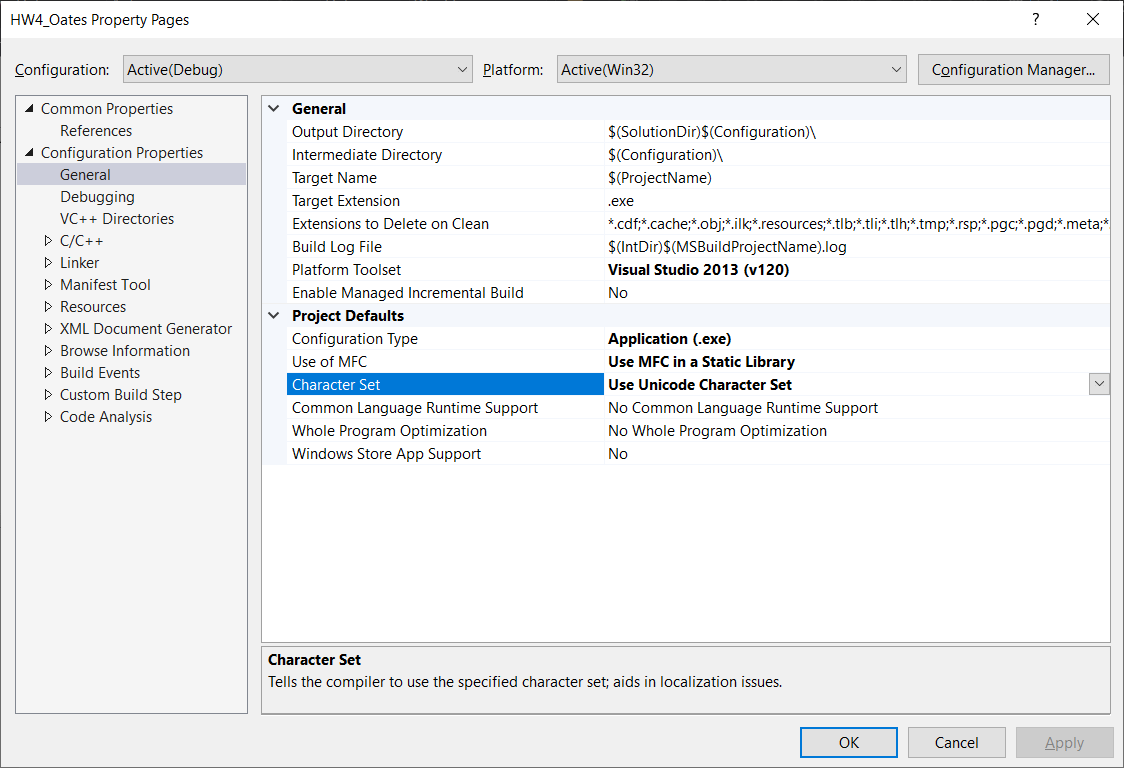


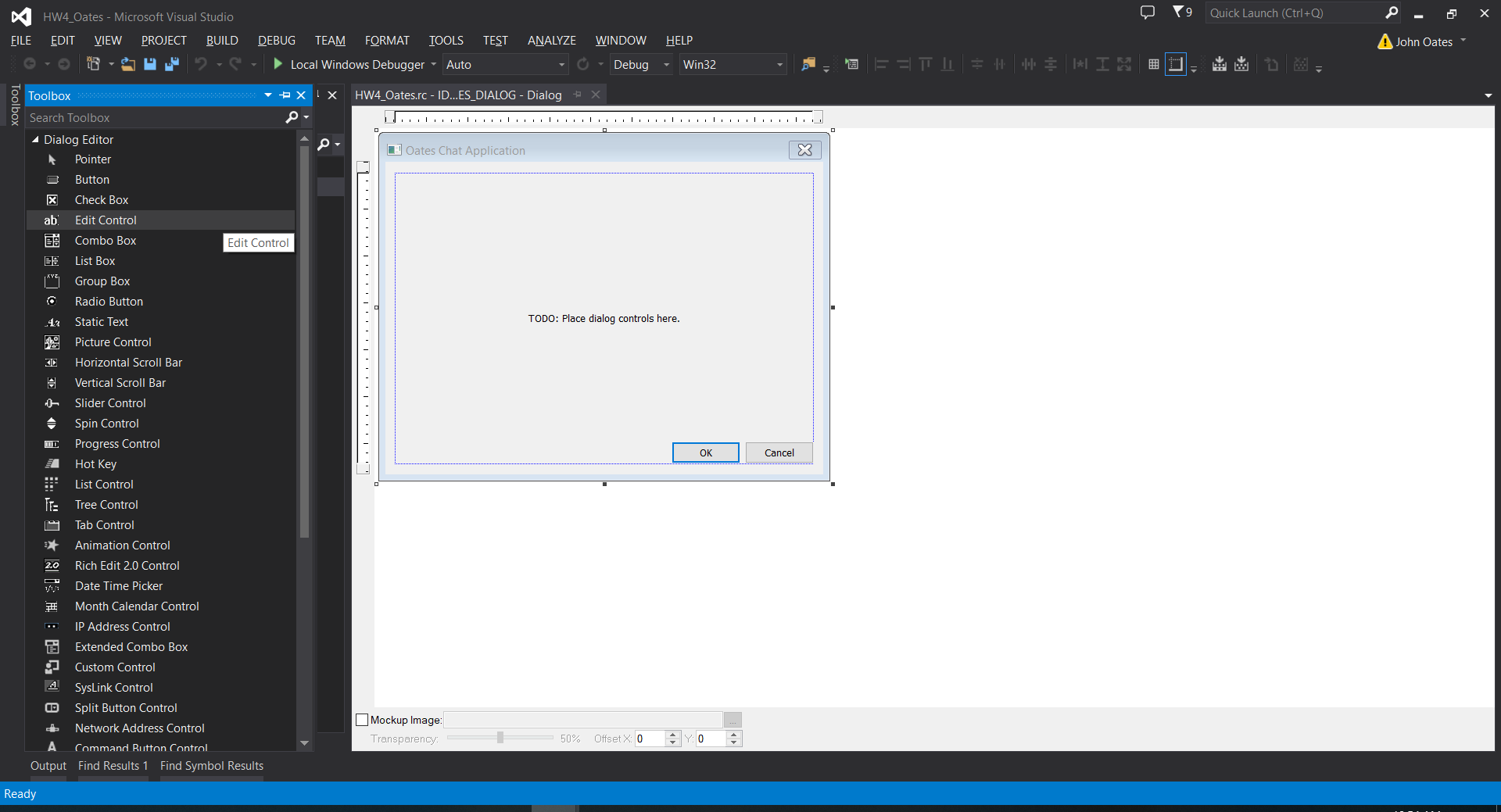
1. The next window lets you set advanced features. Match my selections and press Next.



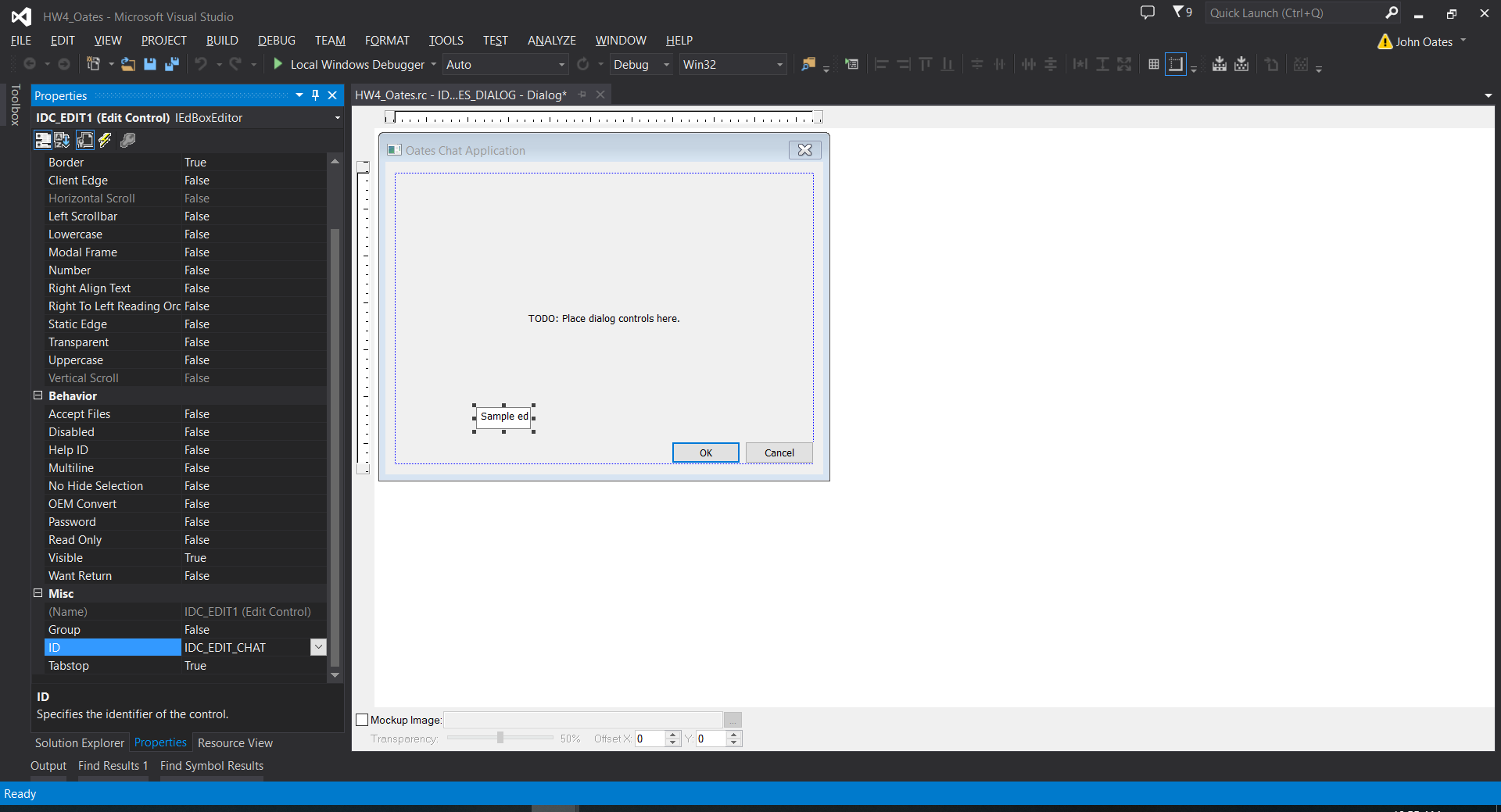
1. Finally, the next window will show you the names of the generated classes (MFC will do much of the work for you in setting up the GUI). You may change the class names if you want, or leave them as they are. Press Finish. (Sorry, I pressed finish before getting a screen shot)
2. The next screen is what you will see when you are done setting up the project.



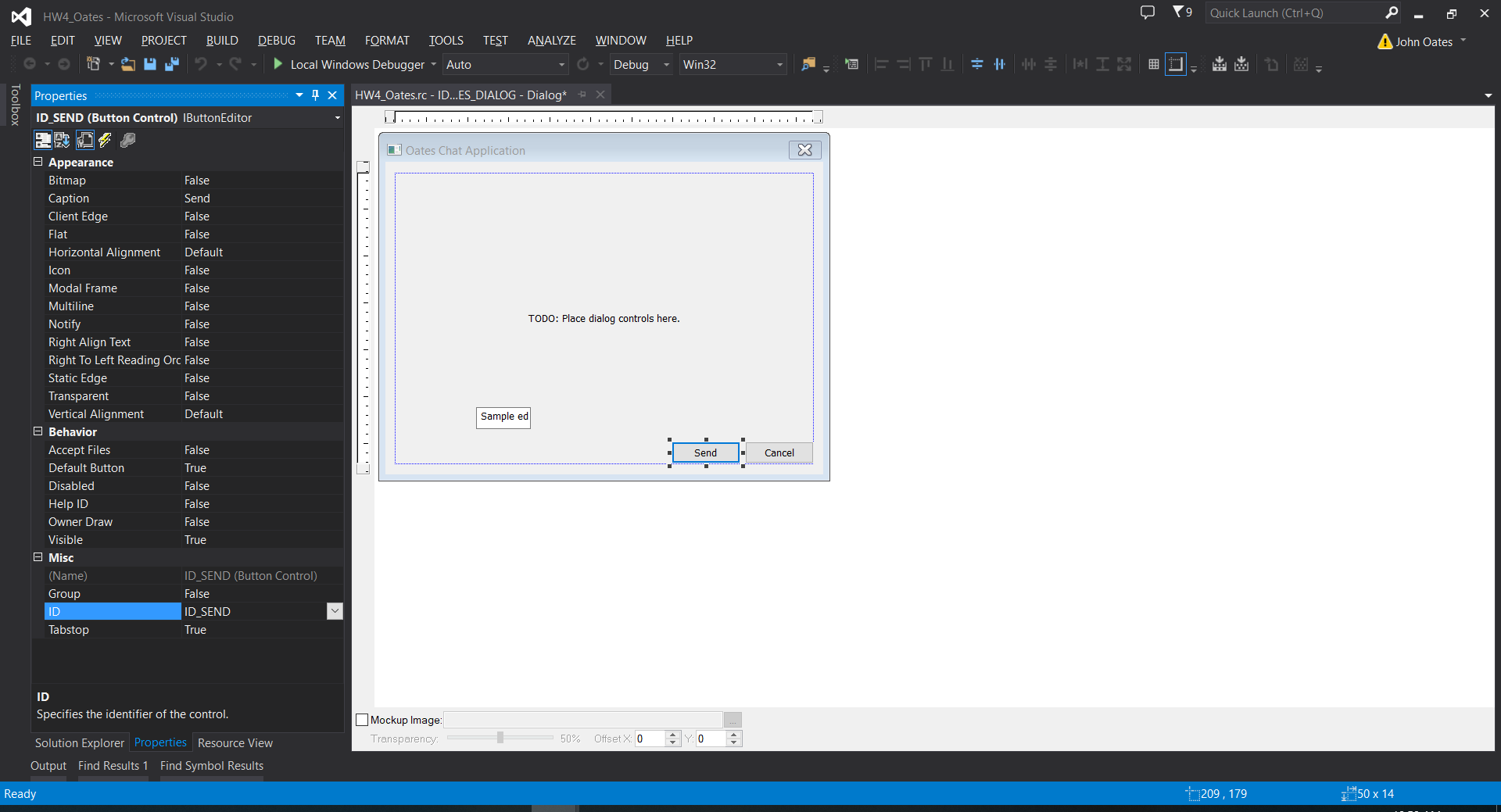
1. The files you will be working in mostly are HW4\_OatesDlg.h, HW4\_OatesDlg.cpp and the Dialog Resource file you see in the screen above. I circled the Toolbox tab. That is where you will find the controls you can drag onto the Chat Application dialog. Yours may be in a different location.
2. The first thing I recommend doing, however, is to go to Project > Properties and change the Character Set from Unicode to Multi-Byte. 
3. Next, make the edit box where you will type in your char messages to send. Open the Toolbox, find Edit Control, and click/drag it onto your Dialog box.



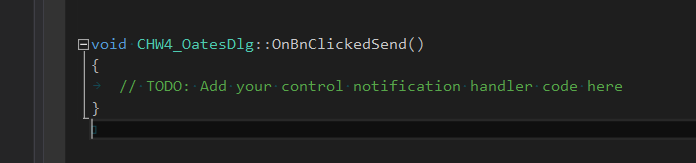
1. Now that you have the control on the dialog, you can move it and resize it to your liking. You also have to give it a unique ID so that the code can reference it. Go to the Properties page (or right click the control and select Properties), and change the default ID (IDC\_EDIT1) to something else. I called mine IDC\_EDIT\_CHAT. You can keep it IDC\_EDIT1 if you want, but in a project with lots of controls, it’s better to give it a name you’ll remember later.



1. Before doing anything else, I recommend changing the OK button to a Send button. To do that, right click the OK button and select Properties. Change the Caption from “OK” to “Send.” And change the ID from IDOK (which has special meaning to MFC) to something like “IDC\_SEND.”



1. Now, you can edit the code. The best way to start doing that is by double clicking the Send button. Doing that will automatically create a method called OnBnClickedSend() in your dialog class. It creates the function prototype, the function definition, and it “wires” the click event to the underlying MFC libraries so that when the application is being run, and the user clicks the send button, the function OnBnClickedSend() will get called. By the way, that means you have already created a multithreaded application, because in order to react to a user event, there must be a separate thread in the waiting state while the main application thread is executing.

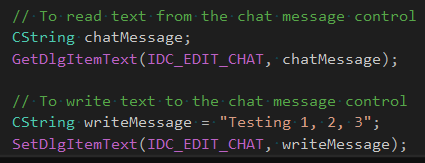


1. Add this line of code where it says “TODO:…”

AfxMessageBox("Hello world!");

Now run your application. Click the Send button and make sure the Hello World message box appears.

1. The only thing left to show is how to read and write to your edit controls. There are several ways to do it – and Google will help you with this – but here’s my preferred method.



This should be enough information to get you well on your way to making the GUI application. I will leave the socket creation, reading and writing up to you for now. As usual, there is more than one way to use a socket, so I will let you do some digging to find your preferred method. Just remember to use UDP. That is, no listen, connect, accept or server/client relationship. Just two peers sending messages to each other. Good luck!