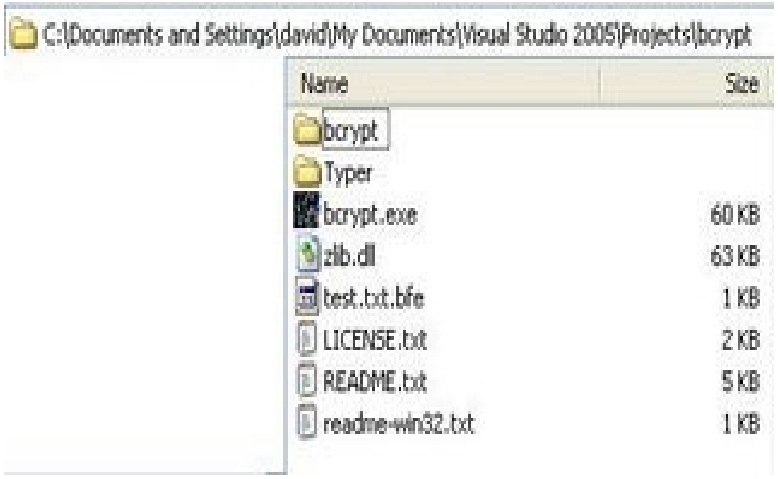
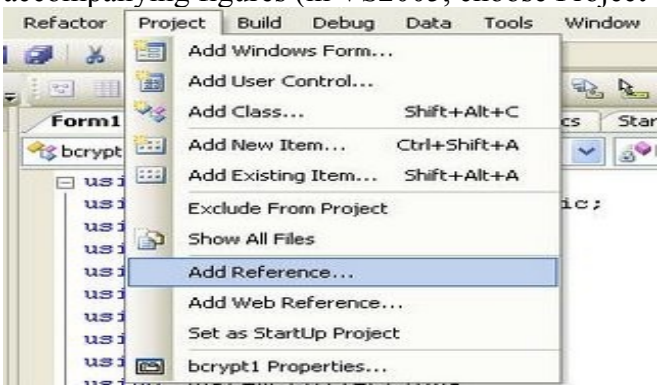
Namespace Usage Relative to Directory Structure for Preexisting C# Projects

We have a preexisting project, downloaded from the internet (www.codeproject.com/csharp/passworddialog.asp), in which a form for obtaining passwords had been used.

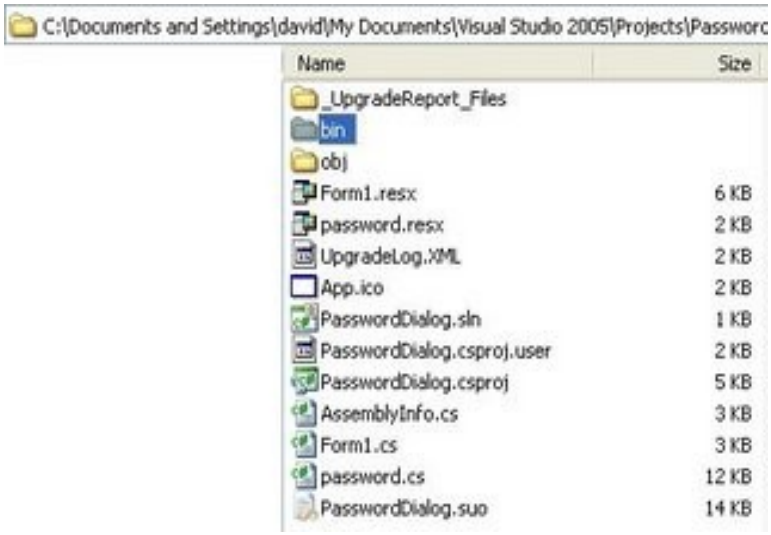
Our project has the directory structure:(since images aren't showing properly, temporarily I'll add the actual directory (C:\Documents and Settings\My Documents\Visual Studio 2005\Projects\bcrypt) which contains a sub-folder called bcrypt, an executable (bcrypt.exe), a dll (zlib.dll) and various readme files and test text files.):



Since the existing PasswordDialog form was well done, and I wanted to use it in this program (the one being discussed here in future publications), but could not find out how to reuse the code without copying it into the then current project (this one). There had to be a better way, and there was. It turns out that you can connect a preexisting project to your current one as shown, partially, in the accompanying figures (in VS2005, choose Project->Add Reference):

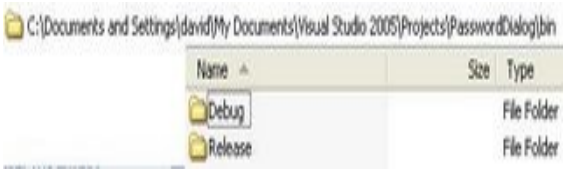


1. First, you attempt to add a reference(...\Projects\PasswordDialog):
2. Inside this directory is the bin Directory



, and

3. Inside this is the Release Directory(...PasswordDialog\bin\Release):



We added the .exe version of PasswordDialog found in the ...\Release directory. Clearly there are other ways of doing this, but this one worked (and that ain't hay).

You will notice that once you've "added a reference", in this case an ".exe" file, that the code in this project can be used provided a "using PasswordDialog;" command has been included (or one explicitly names the procedure as "PasswordDialog.passwordDlg") as shown in the accompanying code text:

==========C# coding follows:

```
passwordDlg passForm = new passwordDlg();
if (passForm.ShowDialog(this) == DialogResult.OK) {
// OK, do something with the password. This is just a     demo.. so lets just
// splash it up for the world to see.
/*MessageBox.Show(this,
"The password entered was: " + passForm.getPass(),     "Password Result",
MessageBoxButtons.OK, MessageBoxIcon.Information);
used earlier to confirm that entered password was the one     desired
*/
}
passForm.Dispose();
```

==========end of C# coding==========

You will notice that the comments come in two flavors (above). The // comments were by the original author, and I have commented out his/her call to the MessageBox.Show() routine, since I don't want an application which shows the password in clear text available to any snoopy passerby. The next lines of code shows where we are going with this. We have:

==========C# coding follows:

```
if (Regex.IsMatch(passForm.getPass(), regex_psswd)) {
//do whatever
}
```

==========end of C# coding==========

We are checking that the password obtained by the dialog is "strong" enough, as specified in the regex_psswd string variable. To see where all this is used, you will have to look at the next "paper" in this series of notes, in which bcrypt.exe is given a windows front-end thereby avoiding the command line interface that it comes with.