Creating and Class Libraries in the Germani/Duffy Applications

3. Creating and Using Assemblies

This exercise is a continuation of the discussion that we introduced in section 11.2.1. In particular, we wish to encapsulate common functionality in assemblies. We create these assemblies from Class Library projects that contain the source code and resources that clients can use. In the interest of simplicity we define a unique (nested) namespace for each reusable assembly that we wish to create. We have already proposed the names for these namespaces in section 11.2.1. The objective is to create assemblies for 1) vectors and matrices, 2) dates and related functionality, 3) Interpolators and 4) Excel Visualisation package. The source code that you need is in the appropriate subdirectory of the Utilities directory on the software distribution medium. Concentrate on one of the above and work out the steps from beginning to end.

Answer the following questions:

a) Create a Class Library project. Give it a name and place the source code and resources that are needed for it to compile and build.

b) The project may depend on other assemblies. For example, an assembly for interpolation will need the assembly containing vectors and matrices. Add all dependent assemblies to the current assembly (use the Add Reference in the project settings).

c) Build the project and make sure that there are no compiler or linker errors.

d) We now wish to test the functionality in the assembly. To this end, we create a Console project containing application code to test the functionality from the assembly. You can create a private assembly – in which case the dll is in the same directory – or by placing the dll in the GAC (Global Assembly Cache). In the latter case the assembly needs a strong name and it needs to be signed.

//

In the current case we have 3 class libraries for 1) matrices, 2) FDM and 3) Excel Visualisation. The corresponding .dlls in the bin\debug directory. When creating a Console application you should add these .dlls by the Add Referece option.

And make all classes *public* (default is *internal* which does not work across dll boundaries).