Chapter 17
Using Libraries and Dynamic Link Libraries

Objective

In this lesson, you will be introduced to the benefits of sharing code among your applications using libraries and dynamic link libraries (DLLs). Not only will you learn how to create a library and a DLL, as well as the circumstances in which to use each, but you will be able to produce a .DLL file that can be distributed easily to other programmers or your end users.

Overview

Both libraries and DLLs provide a means of sharing code among your applications. We explore each type of library in the following sections.

Libraries

You can use a library (or *shared* library) in the same way a .LIB file is used in developing DOS applications. Many of your CA-Visual Objects applications may include the same library and share copies of its code. This library is then linked into the executable (.EXE) file when your application is generated.

Although a library is created much like an application, you cannot run a library as a stand-alone application or create an executable (.EXE) file from one.

The source code of libraries created in CA-Visual Objects may be distributed by exporting the source code to an Application Export Format (.AEF) file; however, this is not usually desirable. Another option is to create a DLL, which provides more flexibility and several advantages.

Dynamic Link Libraries

A DLL is maintained in an external file distinct from any other application. It contains compiled and linked code that can be called from and shared by many .EXE files. DLLs also allow your applications to make better use of available memory, because they are loaded only once, even when shared by many applications.

Memory Efficiency

Using DLLs, you give your applications more memory for data. An .EXE file owns, and can access, a data area called DGROUP (also referred to as *near data* or *static data*) that typically holds your STATIC variables and character constants. This area is limited to 64 KB. However, using DLLs, you can increase the total data memory available area because each .DLL file also has its own DGROUP area. Then, even though applications share the DLL code, each CA-Visual Objects application gets its own copy of the DLL's DGROUP.

Library Distribution

DLLs are an ideal means for distributing your work to other developers without giving away your source code. When you create a .DLL file from the source code in your repository, a corresponding .AEF file is also generated, defining the *public protocol* for the DLL. The only source code you distribute is the .AEF file, which contains only _DLL declaration statements that point to entities in the .DLL file by name.

Once imported, this .AEF file is created as a library (rather than a DLL) that can be included in the search path of any application needing access to the .DLL file.

Ease of Application Maintenance

DLLs make it easier to maintain your code. Since DLLs are linked only at runtime, you can replace a .DLL file without suffering the consequences of having to relink your .EXE file (provided that the public protocol of the DLL has not changed). You will also save development time, since you do not have to rebuild the entities contained in a DLL whenever you rebuild an application.

Two DLL Types

There are two types of DLLs: CA-Visual Objects and "foreign-hosted." A CA-Visual Objects DLL can be used only with CA-Visual Objects applications, supporting its unique language features. A foreign-hosted DLL is a standard DLL that can also be used by applications written on other platforms (for example, C).

Please refer to Shared Libraries and DLLs, in the Operating Environment topic of the CA-Visual Objects online help, for additional information relating to libraries and DLLs.

Exercise

This exercise demonstrates how to create a library and a DLL from existing code in the South Seas Adventures application. In working through the steps, you will also learn how to generate a .DLL file and use the corresponding .AEF file defining its public protocol as a library.

Creating and Using a Library

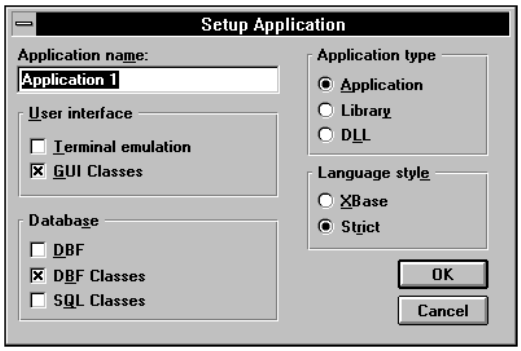
A library is created in much the same way as an application. It does not, however, contain a Start() entity since the code of a library is not intended to be run as a stand-alone application.

In this exercise, you are going to create a library called MyLib. You transfer the IniFileSpec:Class module of the South Seas Adventures application into the library, which can then be used in other applications.

Creating a New Library Application

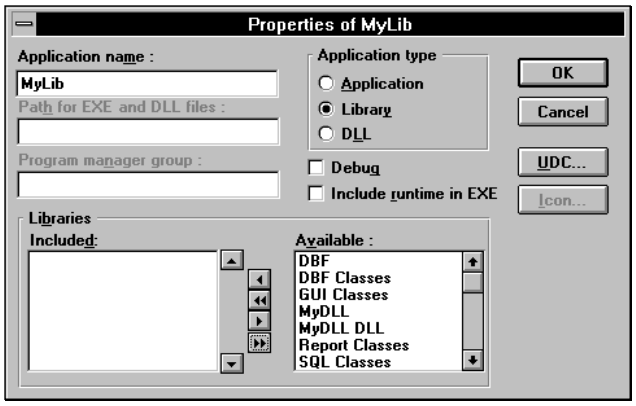
1. From the Application Browser, choose the New Application toolbar button.

   The Setup Application dialog box appears:



2. Type **MyLib** in the Application Name edit control.
3. Uncheck all options in both the User Interface and the Database group boxes.
4. To define the Application Type, click on the Library radio button and choose OK.

   The Properties dialog box is displayed:



   Note that there is no path specified for a library application.

5. Since some of the functions in the new library will make Windows API calls, include the Windows API library in the search path by scrolling through the Available list box and double-clicking on it.
6. Repeat step 5 for the DBF Classes library.
7. Choose OK.

   The empty Module Browser of MyLib is displayed.

Moving Modules Between Applications

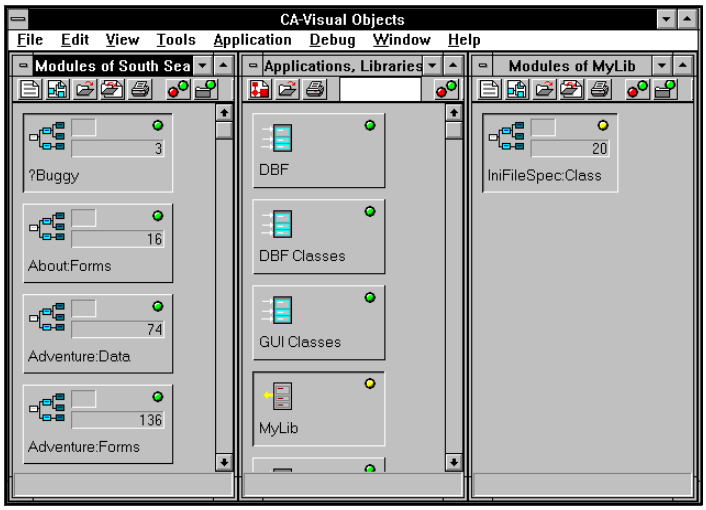
You now need to return to the South Seas Adventures application to move the IniFileSpec:Class module to the Module Browser of MyLib:

1. Return to the Application Browser by selecting Applications, Libraries, and DLLs from the Window menu.
2. Open the South Seas Adventures application by double-clicking on its button in the Application Browser.
3. Select the Tile command from the Window menu.
4. Scroll through the Module Browser, and select the IniFileSpec:Class module by clicking on its button.
5. Select the Move Module command from the File menu.

   The Move Module to Another Application/Library dialog box is displayed.

6. Select MyLib in the To Application drop-down list box, and choose OK.

   The IniFileSpec:Class module is now in the Module Browser of MyLib:



   Notice that all of the modules in the South Seas Adventures application that were using any of the functions in the IniFileSpec:Class module need to be rebuilt. This is because they have lost their link to the functions you moved to the library. You can fix this by including MyLib in the application's search path and rebuilding the library, as shown next.

Building the Library

In order to use the library, you must first build it:

1. Select the MyLib Module Browser by clicking on its window.

2. Choose the Build toolbar button.
3. Clean up the desktop by closing the Module Browser for MyLib, and then choose the Cascade command from the Window menu.

Speeding Up Library Processing

Using a library, you can speed up the generation of an .EXE file and the dynamic execution of applications by creating a prefix (.PFX) file. A .PFX file contains a sorted index of all entities contained in the library. The linker will search through this index when looking for entities not defined in your application.

1. To generate the .PFX file, select the Prefix Lib command from the Application menu.

   The .PFX file is created in the same subdirectory as the CA-Visual Objects repository (for example, C:\CAVO\DATA).

   Note: You should recreate this file whenever you modify a library entity.

2. Close the MyLib Module Browser by double-clicking on its system menu.

The library is now ready to be used in the South Seas Adventures application.

Using the Library

To use a library in an application, you must include it in the application's search path. Let's do this for the South Seas Adventures application, using your new library:

1. Select the South Seas Adventures Module Browser by clicking on its button.

2. Choose the Application Properties toolbar button.

   The Properties dialog box appears.

3. Scroll through the Available list box and select MyLib by double-clicking on it.

4. Choose OK.

5. Rebuild the South Seas Adventures application by clicking on the Build toolbar button again.

The South Seas Adventures application is now ready to be executed and runs just as before. Where it used to access code from its own IniFileSpec:Class module, however, it now accesses code from the library, MyLib.

Creating and Using a DLL

If many of your applications use common code, it would be wise to provide this code as one or more DLLs. At runtime, a .DLL file is shared by the applications that use its code.

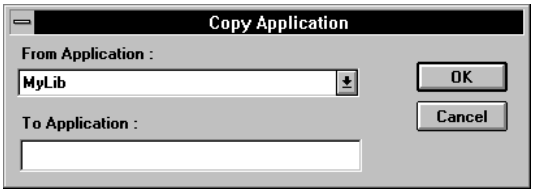
This exercise will take you through the steps to create a DLL from a library and use it from the repository, as well as to create a stand-alone .DLL file, which can be easily distributed and maintained. Lastly, you will learn how to create a foreign-hosted DLL to be used with non-CA-Visual Objects applications.

Creating a New DLL Application

You are now going to create a DLL from a copy of the MyLib library, as follows:

1. From the Application Browser, select the MyLib application by clicking on its button.
2. Select the Copy command from the File menu.

   The Copy Application dialog box appears:



3. Type **MyDLL** as the new application name in the To Application edit control and choose OK.

4. Select the newly created MyDLL application from the Application Browser.

5. Choose the Application Properties toolbar button.

   The Properties dialog box is displayed.

6. To change the Application Type to DLL, select the DLL option and choose OK.

7. Type in the full path for the DLL (for example, C:\CAVO\SAMPLES\SSATUTOR).

8. Choose the Build toolbar button.

You have now created a DLL within the repository, and it is now ready to be used in the South Seas Adventures application.

Using a DLL

At this point, you have created a DLL—although no .DLL file has been generated (a step which comes later in this lesson). You can still, however, use the DLL stored in the repository in the same way that you used the MyLib shared library.

1. From the Application Browser, select the South Seas Adventures application by double-clicking on its button.

2. Choose the Application Properties toolbar button.

   The Properties dialog box is displayed.

3. Scroll through the Included list box and remove MyLib by double-clicking on it.

4. Scroll through the Available list box and select MyDLL by double-clicking on it.

5. Choose OK.

6. Choose the Build toolbar button to recompile the entire application.

The South Seas Adventures application is now ready to be executed and will run just as before. However, where it used to access code from the library, MyLib, it now accesses code from the DLL library,

MyDLL. A few more steps will be necessary to create and use an external .DLL file.

Creating a .DLL File

To use a CA-Visual Objects .DLL file, you must do the following:

- Create the physical .DLL file
- Create a library defining the public protocol for the DLL
- Include the public protocol library in the search path of the application

Let's first create the .DLL file:

1. From the Application Browser, select MyDLL by clicking on its button.

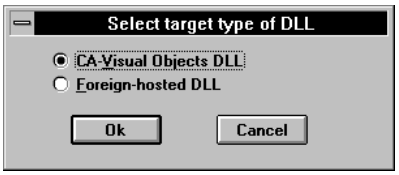
2. Choose the Application Properties toolbar button.

   The Application Properties dialog box appears.

3. Make sure you enter the full path for the .DLL file as the SAMPLES\SSATUTOR subdirectory of your CA-Visual Objects directory, and choose OK.

4. Select the Make DLL command from the Application menu.

   The Select Target Type of DLL dialog box is displayed:



5. The CA-Visual Objects DLL radio button is selected by default. Choose OK to accept the default setting and close the dialog box.

   This creates two files: MYDLL.DLL and MYDLL.AEF. The .DLL file contains the executable code, and the .AEF file contains the prototypes that you will use to access the DLL.

   Both files are written to the path for the .DLL file as specified in the Application Properties dialog box.

6. When the process is complete, close the Module Browser for MyDLL by double-clicking on its system menu.

Tip: At this point, the .DLL and .AEF files can be distributed to other CA-Visual Objects programmers.

Using a CA-Visual Objects .DLL

To have your application access the code stored in a .DLL file, you must define the prototypes to the entities you want to access, similar to the way Windows API library contains the prototypes to the functions available in Windows.

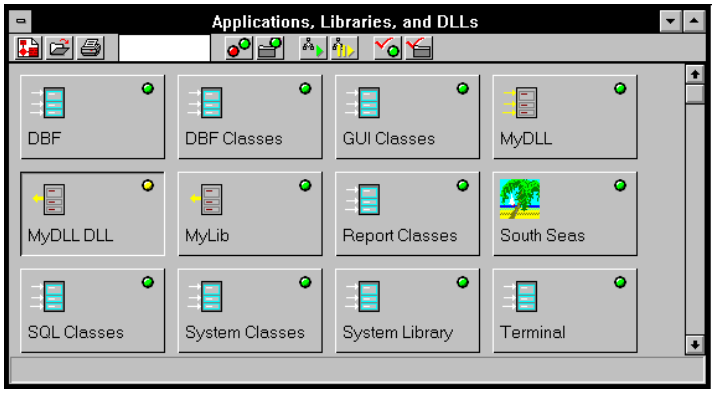
For .DLLs that are created in CA-Visual Objects, this is done for you when you generate the .DLL file. All you have to do is import the generated .AEF file and include it in the application's search path.

Importing Your .AEF file

Let's import the sample MYDLL.AEF file:

1. From the Application Browser, select the Import command from the File menu.
2. Select MYDLL.AEF from the SAMPLES\SSATUTOR subdirectory of your CA-Visual Objects directory, and choose OK.

   A new application is added to the Application Browser, as illustrated below:



The new application, MyDLL DLL, was given a unique name by CA-Visual Objects which added “DLL” to the end of the application name from which it was created (MyDLL).

As its button icon indicates, the new application is a shared library. It contains only the prototypes of the entities in MYDLL.DLL.

3. Choose the Application Properties toolbar button.

   The Properties dialog box appears.

4. Scroll through the Available list box and select the Windows API library by double-clicking on it.

5. Repeat step 4 for the DBF Classes library.

6. Choose OK.

7. Build the new library by selecting the Build toolbar button.

Tip: If there are entities in the DLL that are not intended for direct use by the programmer, their prototypes can be removed from the library. You would then re-export the library as an .AEF file for distribution.

Including Your .DLL in an Application

Finally, to use this new .DLL file, you must include the library which contains its prototypes in your application search path.

Let’s include your new library in the South Seas Adventures application:

1. From the Application Browser, select the South Seas Adventures application by clicking on its button.

2. Choose the Application Properties toolbar button.

   The Properties dialog box appears.

3. Scroll through the Included list box and remove MyDLL by double-clicking on it.

4. Scroll through the Available list box and select MyDLL DLL by double-clicking on it.

5. Choose OK.

6. Select the Build toolbar button to rebuild the revised South Seas Adventures application which includes the new library.

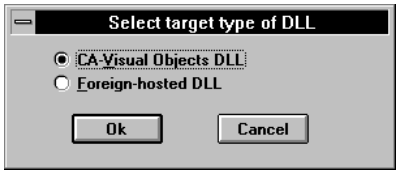
From now on, South Seas Adventures will use the external .DLL file and not the one in your repository. Therefore, any change to the repository source code must be followed by generating the .DLL/.AEF pair and importing the .AEF library once again. Otherwise, the dependent application will not recognize the change.

Creating and Using a Foreign-Hosted DLL

You can create a foreign-hosted DLL simply by following these steps:

1. Select the Make DLL command from the Application menu.

   The Select a Target Type of DLL dialog box is displayed:



2. Click on the Foreign-hosted DLL radio button and choose OK.

However, in the case of the MyDLL library, an attempt to generate a foreign-hosted DLL would result in an error. This is because the MyDLL library does not abide by the rules set out for foreign-hosted DLLs.

Much care must be taken when creating foreign-hosted .DLLs for any platform. Please refer to Foreign-Hosted DLLs, in the Operating Environment topic of the CA-Visual Objects online help, for exact details on the “do’s and don’ts” of creating and using foreign-hosted DLLs.

Summary

In this lesson, you have learned how use libraries and dynamic link libraries to make your applications more efficient. First, you learned to share code between applications by using libraries. You accomplished this by moving a module's code into a shared library and making use of it in an application. The next step was to link this library into your application.

Finally, you learned how to create DLLs, so that your code is shared at runtime. You also learned how to create and use an external .DLL file so that your code can be developed with the most flexibility and portability.

In the next lesson you will create installation disks, and learn how to use Install Maker and CA-Installer for the proper distribution of you CA-Visual Objects applications.