Chapter 2
Exploring the CA-Visual Objects Integrated Development Environment

Objective

This chapter demonstrates how to use the CA-Visual Objects IDE to first import the South Seas Adventures application and then view its components. You will also learn how to build and run the application.

In addition, several CA-Visual Objects application design techniques will be discussed as follows:

- How to best organize files on your hard drive
- How to name your modules
- How to group entities into modules

Overview

Before you begin working on the tutorial exercises in this guide, there are some fundamental concepts that you should know about the way the components of a CA-Visual Objects application are organized and designed.

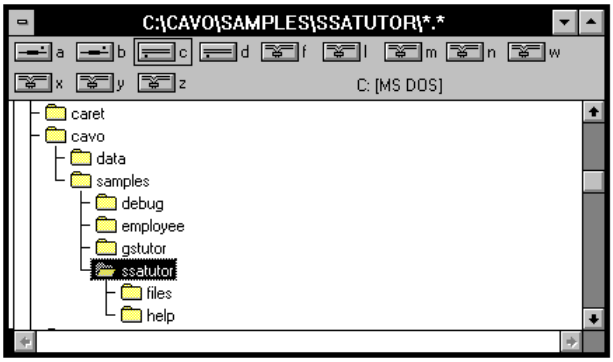
As a programmer, you know the importance of working with a development environment that is easy to manage. You must design a directory structure, define modules that hold all application entities, and be able to access them quickly.

Many of the details that commonly make your life difficult in other programming environments, are automatically handled in CA-Visual Objects.

Choosing a Directory Structure

This section describes how the files related to the South Seas Adventures application are organized on your hard disk. Creating a separate subdirectory allows you to easily separate the various application components when you create install disks and backup sets. For the purposes of this tutorial application, a distinct subdirectory has been created. A more detailed description of the directory structure is presented later in this guide.

For example, the South Seas Adventures application was placed in a subdirectory called \SAMPLES\SSATUTOR. If you have installed CA-Visual Objects to C:\CAVO, the File Manager will display the following directory structure:



The South Seas Directory Structure

This section describes each subdirectory within the South Seas Adventures application development environment.

SSATUTOR

This is the main runtime subdirectory. Any files necessary to run the application will be placed here, such as:

- The application executable, DLL, and help files (.EXE, .DLL, and .HLP, respectively)
- CA-RET report files (.RET)
- Data files (for example, .DBF and .NTX)

SSATUTOR\FILES

This subdirectory is for resource files, such as bitmap (.BMP), icon (.ICO) and cursor (.CUR) files. Since these files are included in the executable file (.EXE or .DLL), your users do not really need the files in this subdirectory. You can also use this area to keep any application (.AEF) and module (.MEF) export files.

SSATUTOR\HELP

Creating context-sensitive help for the application is a project in itself. It can be easily segregated from the main development area. See the "Adding Help to Your Application" chapter for more information on creating help. The actual help file (SSA.HLP) associated with the application will be placed in the \SSATUTOR subdirectory.

The reasons for using this directory structure for the South Seas Adventures application are:

- The application development directory structure resembles a user's directory structure when the system is installed—resulting in a similar environment.
- It is easy to create an installation set since the runtime files are kept in one place. The only other files needed by a user to run the application are the DLLs required by CA-Visual Objects. (For more information on DLLs, see the "Distributing Your Application" chapter in this guide. Also, see Appendix A, "Creating a Path-Independent Application.")

Creating Path-Independent Applications

CA-Visual Objects runtime file handling depends on the location of certain files (including database, report, and help files), as well as how you specify their locations when you create the server, report, and help building blocks.

To the extent possible, you should refrain from either specifying or hard coding any path information for these building blocks in order to maintain drive and directory independence for your application. There are also several steps you need to perform when you design entities using the CA-Visual Objects visual editors.

It is recommended that you refer to Appendix A, “Creating a Path-Independent Application,” for more information related to path-independent applications.

The CA-Visual Objects Repository

Many of the traditional development details are handled by the CA-Visual Objects repository. The repository stores and organizes all of the application building blocks (or entities). It also manages other details of your application, including:

- Source code files
- Compiled code files
- Binary design components (such as windows and menus)
- Compiler options
- Directory information
- Dependency information

You no longer have to deal with such details as the location of your include (.CH), source code (.PRG), make (.RMK), or object (.OBJ) files. When you save an application, all the relevant information is stored in the repository files. This information can only be viewed or utilized while you are working in the CA-Visual Objects IDE.

The repository provides you with a three-level hierarchical, object-oriented view of an application. It allows you easy access to the many entities that define the application.

Application Component Hierarchy

Application Level

The *application* is the top level in the hierarchy. GLOBAL variables are visible across the entire application.

Module Level

Modules exist on the second level of the hierarchy. Modules are similar in nature to traditional source code files, since they contain and encapsulate the supporting entities. STATIC GLOBAL variables are visible only within the module.

Entity Level

Entities are the smallest identifiable components of your application and are analogous to the functions, procedures, constants, and class definitions of traditional source files. The CA-Visual Objects repository manages each entity as a separate unit. LOCAL variables are visible to the creating entities only.

Automated Make and Entity-Level Compiling

The repository also keeps cross-reference information between the entities of your application. When you build your application, CA-Visual Objects knows which entities it has to compile, based on the changes you have made—only the entities that have changed will get recompiled. This is known as “entity-level” incremental compiling. Hence, you no longer need a traditional “make” file.

Grouping Your Entities into Modules

Module Design Considerations

As a programmer, you may often find yourself wondering how many modules are appropriate for the given application. There are a few general rules that may be helpful in this process, as described below:

1. The IDE visual editors place all source code entities related to a primary design entity in the same module as the design entity.
2. To avoid confusion, do not mix data servers, menus, and windows in the same module.
3. To avoid having too many entities in a module, restrict a module so that it contains no more than two or three windows or one menu, if possible. Although a question of personal choice, most modules should have between 20 and 100 entities.
4. Where possible, place all custom source-code entities (developer-coded) in a module other than the one in which the window or menu resides. You may have to cut and paste source code between modules to accomplish this.
5. Use descriptive two-part names for the modules. Do not make the names too long; otherwise, the size of the buttons in the module browser will become excessively wide.

In the South Seas Adventures application, many of the module names begin with the name of one of the business data types: adventure, customer, employee, invoice, item, and payment. The second part of the module name is a word such as class, data, forms, menu, methods, or reports. This naming convention indicates what is contained in the module. Note that the developer-coded methods are placed in the Methods modules.

Module Naming Conventions

This section discusses some of the tutorial modules in depth in order to demonstrate how the design considerations were applied. What you are striving for in your naming convention is ease of management in the future. Take advantage of long names for modules as well as entities. This will make the components of your application easier to recognize.

The names of the South Seas Adventures application modules are all two-part names separated by a colon (for example, Adventure:Data). The first part defines the primary focus of the entities in the module and the second part identifies what the entities are used for. For example, ":Forms" indicates that the module contains data windows, dialog windows, or a shell window.

To give you a practical idea of how these naming conventions are applied, let's now examine some of the Adventure modules:

Adventure:Data

This module contains all entities related to the two data servers associated directly with an adventure, the adventure header file (ADVHDR.DBF) and the adventure detail file (ADVDTL.DBF).

Adventure:Forms

This module contains all the entities required for the adventure forms. The Window Editor was used to create these forms; therefore, the entities of the most concern within this module are naturally the window entities. There are actually five forms (window entities) in this module that let you add, edit, or browse data in the data servers mentioned above. These forms are named as follows:

- AdventureBrowser
- AdventureDetailSubform
- AdventureSubform
- EditAdventureWindow
- NewAdventureWindow

	The AdventureSubform is a tabular display used on the AdventureBrowser window, and the AdventureDetailSubform is a tabular display of the items in the master-detail EditAdventureWindow.
Adventure:Methods	This module contains all the methods that were not generated automatically by the CA-Visual Objects editors. This code was separated from the generated code to make it easier to locate.
Adventure:Reports	This module contains a report entity as well as a window entity. The window entity is in this module because it is a dialog box that defines the scope of the report that will be created.

In addition to modules that relate to one of the six business data types, you will also need some application-related modules. These contain entities used in several places. In the South Seas Adventures application, the general purpose modules include:

Password:Forms	This module contains the password dialog window and related entities.
SSAChild:Menu	This module contains the menu attached to the child windows.
SSAShell:Forms	This module contains the shell window (SSAWindow) for the application.
SSAShell:Menu	This module contains the menu attached to the shell window.
App:Misc	This module contains miscellaneous entities that can become a library or a DLL.
App:Resources	This module contains any cursors, bitmaps, and icons that the application needs.
App:Start	This module contains the application Start() method and any global entities for the application.

Now that you know the basics about how application components are handled and designed, you are ready for a hands-on tour of the IDE. You will discover the ways it can help you work with your application—during various stages of development.

Exercise

In the following exercise, you will open the CA-Visual Objects Application Browser, import the South Seas Adventures application, set some application properties, compile the application, and then run it.

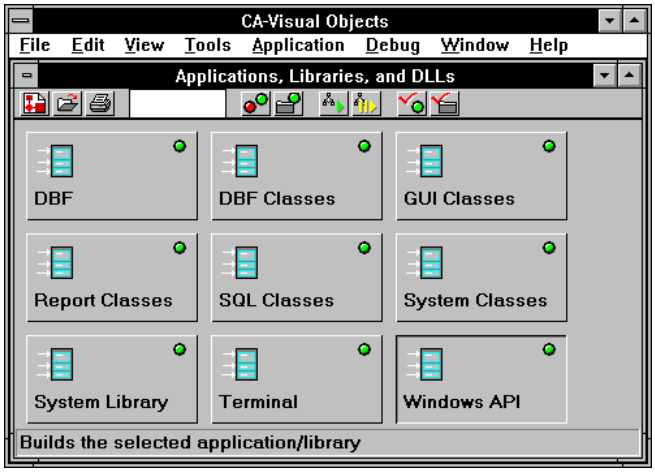
Using the Application Browser

Browsers provide you with a logical way to access and view the components of your application. There are three browsers in CA-Visual Objects, one for each level in the repository hierarchy (application, module, and entity).

Let's look at the Application Browser by performing the following steps:

1. Start CA-Visual Objects from the program manager by double-clicking on the CA-Visual Objects program icon.

   The Application Browser, the first in the series of browsers, appears on the desktop:



Initially, the Application Browser displays all of the libraries supplied with CA-Visual Objects. Each library is represented by a button in this window. Similarly, the applications and libraries that you create also appear as buttons in this window.

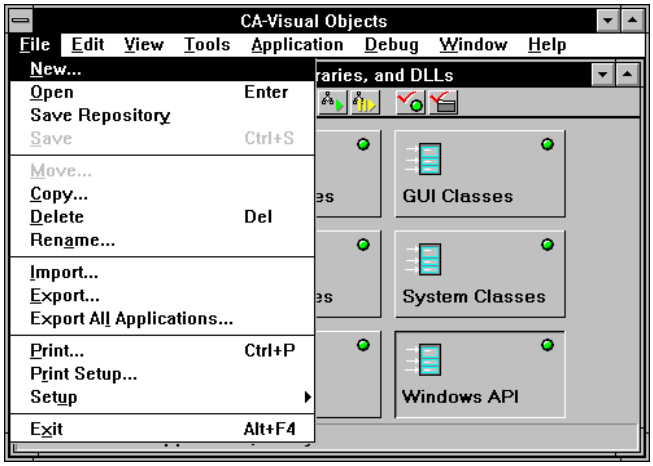
2. Notice the CA-Visual Objects default icons that are displayed on the buttons in the Application Browser. These distinctive icons help you visually distinguish between applications and libraries.

   The following table shows the default CA-Visual Objects icons:

Icon	Description
	CA-Visual Objects System library icon
	Application icon (which you can create)
	User-defined library icon
	User-defined DLL icon

You can also create your own application icon, which you will learn more about in the “Working with Icons and Cursors” chapter in this guide.

3. The Application Browser provides you with those commands that enable you to manage your applications—New, Open, Copy, Rename, and so on. Choose the File menu and inspect its pull-down menu to view all of the commands available to you:



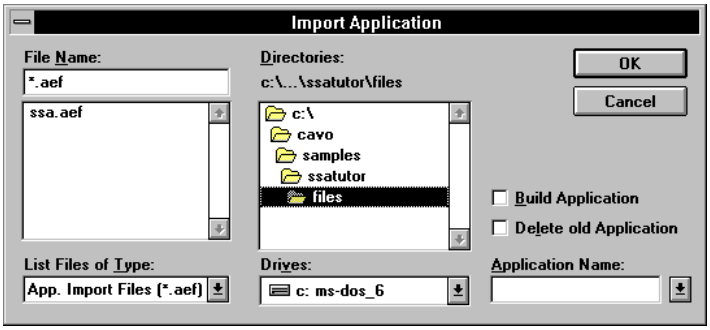
The Import and Export commands are provided to allow you to move applications to and from external disk files, since applications are stored in the repository. You will find these options useful for trading applications with other developers and for backup purposes.

Importing the Application

Let's begin by importing the South Seas Adventures application which is used throughout the remainder of this tutorial.

1. Choose the Import command from the File menu.

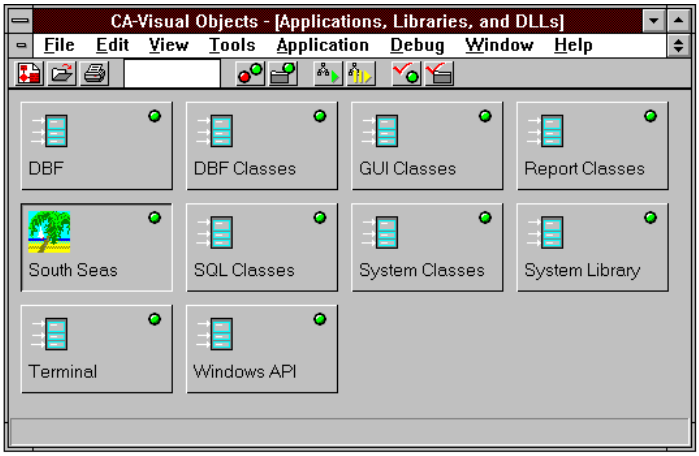
   The Import Application dialog box appears:



2. Select SSA.AEF. This is the South Seas Adventures application export file. It can be found in your CA-Visual Objects \SAMPLES\SSATUTOR\FILES subdirectory.

3. Choose OK.

   After the application is imported, a new button appears in the Application Browser. This button represents the South Seas Adventures application, as shown below:



Notice the palm tree icon on the South Seas Adventures application button:

This icon is associated with the application through the Application Properties dialog box, which you will learn about in the next section.

Configuring Your Application Environment

For the purposes of the tutorial, there are a few settings that you may have to modify, based on how you installed CA-Visual Objects. You will also look at the compiler options that are available to you.

Application Properties

As suggested previously, you should create a subdirectory for each application or library that you create. This ensures that your disk stays organized, by keeping files for each application separate, and eliminates confusion about what files belong to a specific application. The path for EXE and DLL files can then be set to the application's subdirectory.

However, it is possible that CA-Visual Objects is installed to a different drive or directory, for instance to D:\CAVO. If so, you must modify one of the high-level properties of the application—namely, the path for the generated EXE and DLL files. You can modify the properties of an application or library at any time during development. The application's Properties dialog box allows you to specify the following:

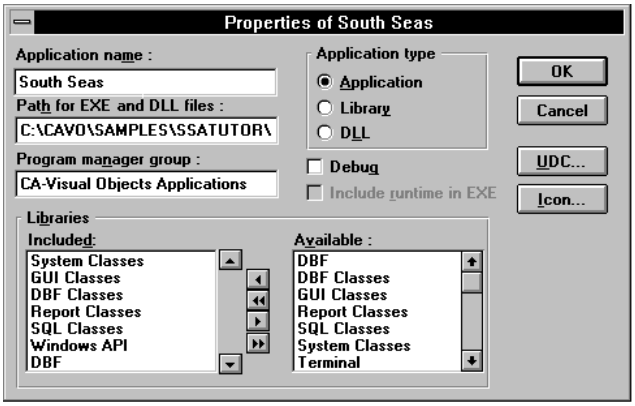
- Application name
- Application type
- Libraries to include in your search path
- Path for a generated file (.EXE or .DLL)

- Program Manager group in which to place the program icon for your generated .EXE file
- Default debugging status
- Whether the generated .EXE or .DLL file should include CA-Visual Objects runtime files

Now, let's see how this applies to the South Seas Adventures application. If you have installed CA-Visual Objects on a drive and directory other than C:\CAVO, you must modify the application properties as follows:

1. Select the Application Properties toolbar button.

   The Properties of South Seas dialog box appears:



2. In the Path for EXE and DLL Files edit control, type the drive and directory to which you installed CA-Visual Objects. For example, if you installed CA-Visual Objects to the D:\CAVO directory, set the path to:

   D:\CAVO\SAMPLES\SSATUTOR\SSA.EXE

3. Choose OK.

   For more information on application properties, see “Browsing Applications, Modules, Entities, and Classes” in the *IDE User Guide*.

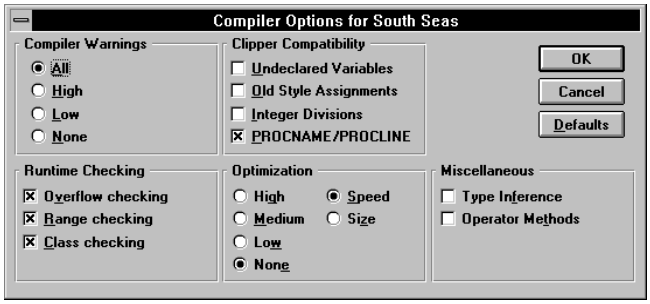
Compiler Options

Before attempting to compile the application, it is wise to review its compiler options. You can specify different compiler options for each application at any time during the development process.

Let’s now look at the compiler options for South Seas Adventures application:

1. Select the Application Compiler Options toolbar button to view the compiler options available to you.

   The Compiler Options dialog box appears:



2. Notice that the default system compiler options, with the exception of All in the Compiler Warnings group, are used for the South Seas Adventures application.

3. Choose OK to close the Compiler Options dialog box.

For more information on compiler options, see the CA-Visual Objects on-line help or “Working in the CA-Visual Objects Desktop” in the *IDE User Guide*.

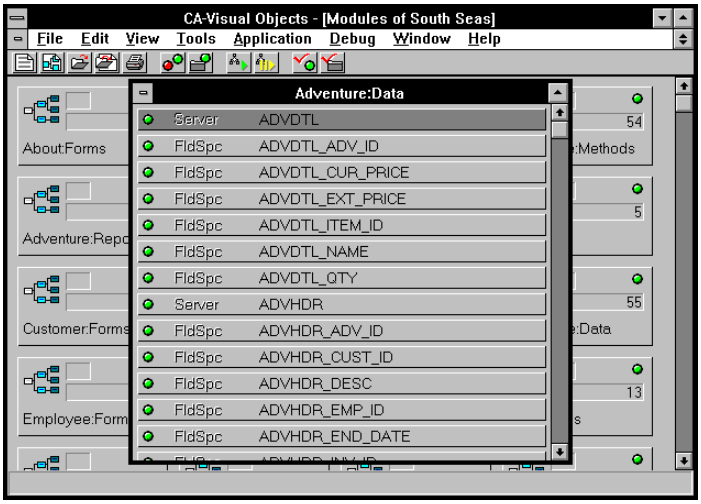
Using the Module and Entity Browsers

In the following exercise, you will open the CA-Visual Objects Module Browser, examine the entities in a module, and use the Entity Browser to look at the various types of entities.

1. Open the South Seas Adventures application by double-clicking on its button in the Application Browser:

2. Double-click on the Adventure:Data module to examine its entities.

   The Entity Browser appears:



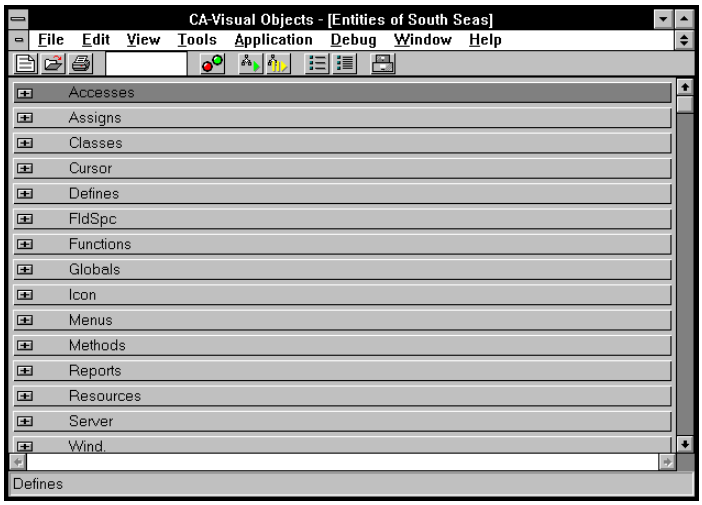
   There are two server entities (AdvHdr and AdvDtl), each associated with many field spec entities. Further down in the list, you will find the two data server classes and related methods, accesses, and assigns.

   Note: If an entity name is surrounded by parentheses, the variable or function is local (for example, STATIC DEFINE).

3. Scroll through the list, and double-click on one of the two Init() methods and a few access and assign entities to see the kind of source code that is generated by the DB Server Editor.
4. Close the Source Code Editor by double-clicking on its system menu.
5. Close the Module Browser by double-clicking on its system menu.
6. Choose the Entity Browser command from the Tools menu.

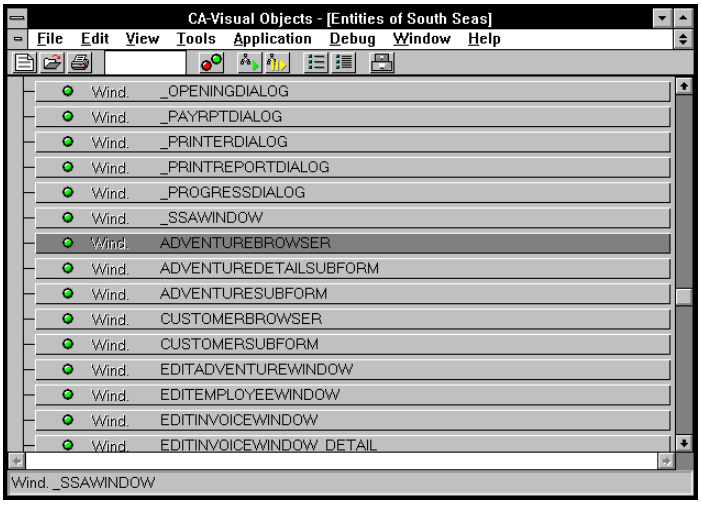
   You can now see all of the access entities, although if you scroll through the entity list, you will see the other entities in the Adventure:Data module.

7. Click on the Collapse All toolbar button to view all of the entity types:



8. Click on the plus button to the left of the Wind. entity label to see all of the window entities.

9. Scroll down until you see the window entity named AdventureBrowser, as shown below:



The windows above it are the dialog windows, while those below it are the other data windows. If you were to double-click on any window entity, a Window Editor session would be started.

10. Click on the Collapse All toolbar button again to return to the prior view.

11. Experiment with some of the other categories, and double-click on some of the entities to examine them.

12. Close the Entity Browser by double-clicking on its system menu.

Using the Class Browser

In the following exercise, you will open the CA-Visual Objects Class Browser, and examine a class and a subclass, including their various types of entities.

1. Select the Class Browser command from the Tools menu.

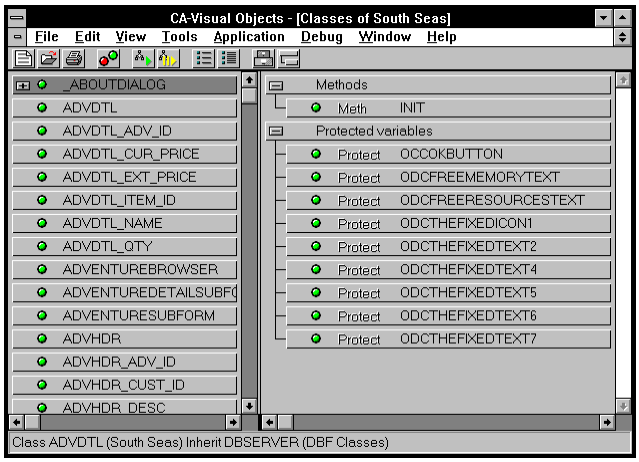
   The first class name shown is _AboutDialog and the second is AdvDtl.

2. Click on the plus button, to the left of the _AboutDialog class.

   The _AboutDialog class name remains highlighted.

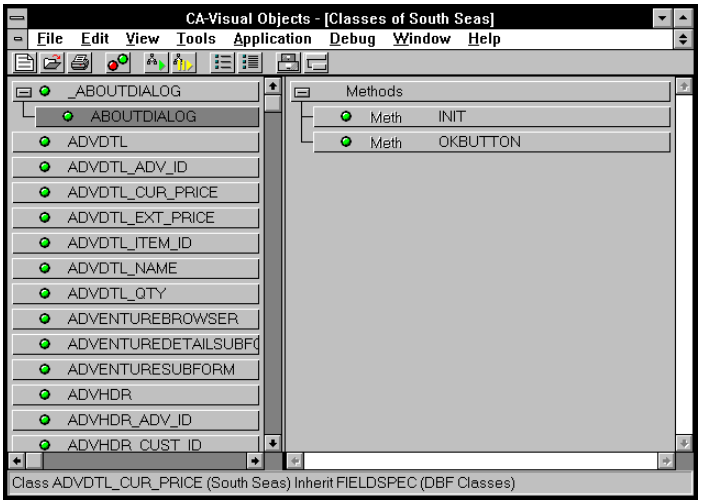
3. Examine the right half of the window.

   You will find an Init() method and nine protected variables:



4. Click on the AboutDialog entity, which is a subclass of _AboutDialog.

   The Class Browser displays the following:



   Note that the right side of the browser now shows only two methods, Init() and OKButton(). These two methods are defined explicitly for the subclass, while all other properties of the parent class are inherited. The subject of inheritance is discussed in greater detail in the “Inheritance and Subclassing” chapter.

5. When you have finished exploring the classes and their related methods, close the Class Browser by double-clicking on its system menu.

Building the Application

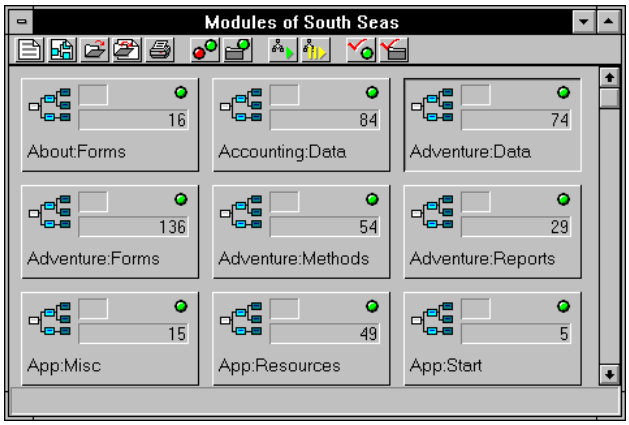
Once you have imported (or finished developing) your application, it can be easily compiled. To build the South Seas Adventures application:

1. Click on the South Seas Adventures application button in the Application Browser:

2. Choose the Build toolbar button.

Note: When you import an application, none of the entities are compiled. The South Seas Adventures application could be considered a medium-sized application, so this initial build could take some time. Subsequent builds will be much faster, since only those entities that need compiling, based on your changes, will be compiled.

Once compilation is completed, you will see "Build Done" displayed on the status bar. The Module Browser appears showing all the modules for the South Seas Adventures application:



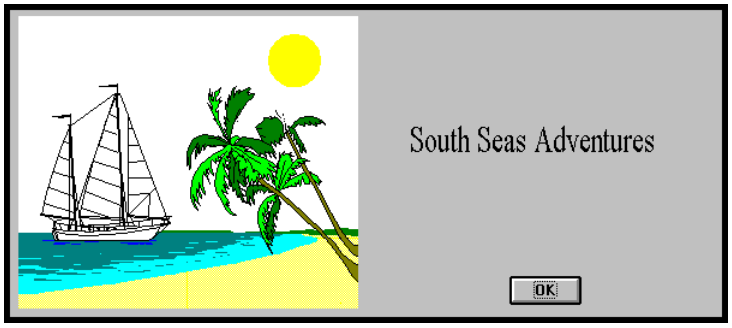
Once your program compiles successfully—indicated by a green LED on the application button—you can run your program. The next section discusses how to do this.

Running the Application

During development, you can either run your application dynamically from the IDE or you can generate an .EXE file and run it from the Program Manager.

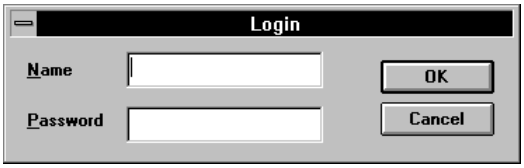
Regardless of the option you choose, when the application is run, the current directory will be set to the path specified in the Application Properties dialog box. This allows you to test both versions from the same directory.

Throughout this tutorial, you will be asked to run (or *execute*) the application. When you do so—by either clicking on the Execute toolbar button (or by selecting the Application Execute menu command)—the following opening dialog box is displayed:



To continue, simply click on the OK button. Because of the nature of the South Seas Adventures application, a login dialog box was created as part of the application design and is required to enter the system.

The Login dialog box appears:



You can enter the system by logging in using the following procedure:

1. Type **User** in the Name edit control.
2. Type **Trainee** in the Password edit control.
3. Choose OK.

Running the Program Dynamically

As you incrementally develop your application, and then compile your latest changes, you can run it without even having to leave your IDE environment. To do this, perform the following steps:

1. Click on the South Seas Adventures application button in the Application Browser:

2. Click on the Execute toolbar button to run the program from the IDE.
3. Choose OK in the South Seas Adventures opening dialog box.
4. Choose Cancel in the Login dialog box to close the application.

   An Error dialog box appears.
5. Choose OK to close the Error dialog box.

Creating and Running an Executable File

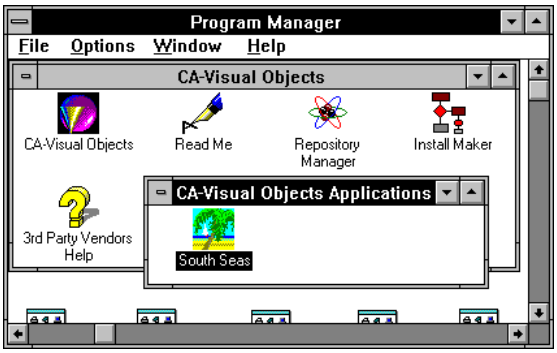
Typically, after you have finished development you will want to create a stand-alone executable program (.EXE) which can be run without the CA-Visual Objects IDE.

To create the .EXE file and run this program, perform the following steps:

1. Click on the South Seas Adventures application button in the Application Browser:

2. Click on the Make EXE toolbar button.

   Once the .EXE file has been generated, the Program Manager will be brought to the foreground. Your application icon will be in the program group specified in the Application Properties dialog box:



3. You may now run the program by double-clicking on the South Seas Adventures application icon.

   The opening dialog box appears.

4. Choose OK.
5. Choose Cancel in the Login dialog box to close the application.

   An Error dialog box appears.

6. Choose OK to close the Error dialog box and return to the Program Manager.
7. Press Alt+Tab to return to CA-Visual Objects.

Summary

This chapter provided a brief overview of the IDE, as well as some useful tips to make your application development easier. If you need more information on using and navigating through the IDE, see the CA-Visual Objects *IDE User Guide*.

You can now move on the next chapter, which shows you how to create and use a data server, an integral part of the South Seas Adventures application.