Chapter 3 VFE's IDE

In this chapter, we will take a brief look at the elements of VFE's Integrated Development Environment. If you think of VFE as consisting of two major pieces—the toolbox and the building materials—the IDE would be the toolbox.

What you find when you look in the VFE toolbox is a plethora of tools to help you assemble the framework classes (which are the building materials) into an application. These tools enhance the VFP tools and are specifically aware of the framework classes.

As you learned in Chapter 1, "The Big Picture," there is a wide range of tools in the IDE. These include the Application Manager, Application Builder, wizards and builders. There are also some tools in the IDE that allow you to modify the behavior of the framework. We will briefly describe those here, and we'll take a closer look at some of them later in the book.

The Application Manager is the tool where you will start a new VFE project. So, let's get started and create our Tutorial application with the Application Manager.

Tutorial

At this point, you should have VFE installed on your computer, including Service Pack 2 and all subsequent fixes and updates published on the VFE Web site.

1. Start Visual FoxPro using your VFP shortcut with your VFE-specific startup program.
2. Start the Application Manager by choosing it from the Express menu, or by typing *assist* in the command window. At this point, the Application Manager should be centered in your VFP desktop. If not, refer to the VFE installation directions to see what might have gone wrong.
3. Press the Create button on the VFE Application Manager. This is the leftmost button with the icon that most applications use to designate "new." You will be presented with the first page of a wizard that will step you through the creation of the application project.
4. Click on the command button with the ellipses to create your project file. You will be presented with the VFP New Project dialog, with C:\VFE6 as the current directory. At this point, we will use the functionality of the New File dialog to create a directory in which to store our project, prior to creating the project.
5. Navigate up to the root directory of C:, and then create a new directory named "VFEBook." (If you have already copied the Developer Download files to your PC, the VFEBook directory will already exist on your C: drive; just navigate into it.)
6. Add a directory named "Tutorial" in the VFEBook directory. This will be the "root" directory for your application.

7. Double-click into the Tutorial directory.
8. In the file name field of the dialog, type “booktutorial” as the name of your project, and then click the Save button. You will then be brought back to the wizard, which will have your new project name with the path you created entered for you.
9. Press the Next button on the wizard, which is indicated by the right arrow icon. You will be presented with Step 2.
10. This step presents you with some directory information. We will use all the defaults except for one. Highlight “Intermediate Class Libraries” and press the Select Directory button, or just double-click the selection in the list. You will be presented with the Enter Directory dialog.
11. It would be nice if we were given an ellipses button here, but since we’re not, we will have to do it the hard way. Enter “C:\VFEBook\Tutorial\iLibs” in this text box and press the OK button. An information dialog will be presented—read the section “iLibs” later in this chapter for more information on this. Click OK.
12. Press the Next button of the wizard, and you will be presented with Step 3.
13. Enter the information requested in this dialog. We used “Book Tutorial” for the application name. This information will be used to create a splash screen. Also, the name entered here is the name that will be displayed in the Application Manager. The “use local data” switch will be discussed in Chapter 19, “Client/Server.” (Remember, you already defined the project/exe name in the first step.)
14. Press the Next button to move to Step 4. Enter the requested information. We left this section blank because we didn’t have these files. You can always add these to your application at a later date.
15. Press the Next button on the wizard to move to Step 5. Select the defaults for this step, and then press the Next button to move to Step 6.
16. *Yeah!* VFE is ready to take what it learned in this interrogation and create your project directory structure and a starter or shell application. Press the Finish button.

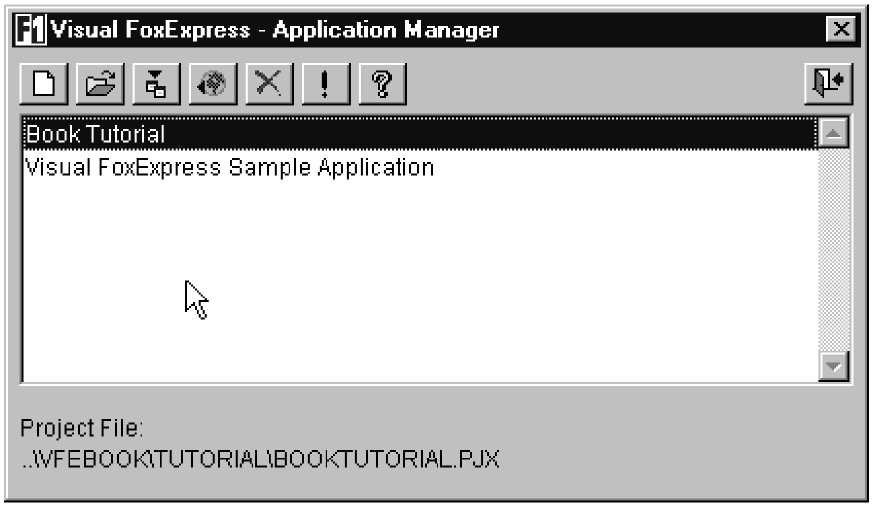
After much whirring, buzzing, messages and status bars, VFE will complete its task. Your project has been created, and you will now be presented with the Application Builder, which is the main interface you will use to build your VFE application. At this point, you can even run your application following two simple steps:

1. Press the Build button. Make sure the “Build Project” action is selected, and then click the OK button.
2. Press the Run button. With a little luck, your application will run. The Options dialog will be present the first time you run a new app; just press OK, and then you can look around the menu and see which standard forms are already built for you.

For more information on your new application, see Chapter 9, “Your VFP Application.” When you’re done looking around, choose File | Exit from the menu.

Application Manager

As you can see in **Figure 1**, the Application Manager has eight buttons on it, which represent the functionality of this tool. The VFE documentation does a fine job of describing the basic function of each button; see the VFE documentation topic “The Visual FoxExpress Application Manager.”



Book Tutorial
Visual FoxExpress Sample Application

Project File:
..\VFEBOOK\TUTORIAL\BOOKTUTORIAL.PJX

***Figure 1**. The VFE Application Manager.*

By far the most important function of the Application Manager is new project creation. Now we’ll explain why we made some of the choices we did when we ran the New Project wizard.

Project directory

If you followed the VFE tutorials, you were prompted to create a directory in a subdirectory beneath VFE6 to hold your VFP projects. While this approach works, we prefer to create a directory outside of the VFE directory to hold application projects.

The main reason for this decision is to allow for the possibility of something becoming corrupted or otherwise messed up inside the VFE6 directory. For example, what if you installed the service pack without telling WinZip to expand folders? All your files would have been placed into the VFE6 directory. Or, if you specified the wrong folder to unzip to, all the files would be in the incorrect directories.

In a situation like this, it would be much simpler to delete the VFE6 directory and reinstall from scratch. This is similar to installing VFP in the program files directory and building applications in a separate directory. Also, if you follow the common directory structure, you should be able to move your applications to a different directory, as there will be no direct references to the framework classes in your application classes.

iLibs

While most of the classes discussed here start with “C” and are in the …\VFEFramework\Libs directory, the actual classes you use to build your application are the classes that start with “I” for intermediate.

The I-classes or I-layer give you a place to modify the functionality of the shipped VFE classes and still not lose or have to modify the shipped code. When creating applications, you can have all applications share a common set of iLibs, or you can create a new set for each application. Also, you could use a common set for some apps and another common set for other apps.

The reason you created a separate set for this application is because when you get to Chapter 16, “Extending the Framework,” you may not want these changes in your apps—there won’t be any conflicts because they wouldn’t use these specific iLibs.

What gets created?

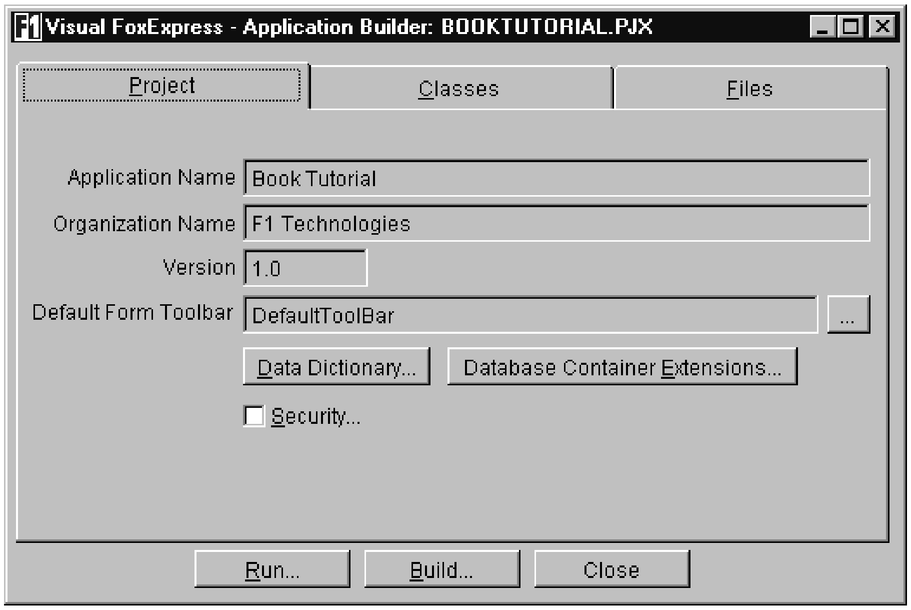
During all that whirring and buzzing, VFE is creating your new project and an application directory structure. For the most part the directories are empty, but some files are placed there. Let’s see what directories are created and why.

1. A VFP project file is created. This project file is populated with all of the class libraries of the framework. VFE also creates some empty class libraries for you to store your classes in.
2. The following subdirectories are created under your project directory:
    a. *Data*—This is the directory where you are expected to store your database and table files.
    b. *Libs*—This is the directory where you are expected to store your class libraries.
    c. *iLibs*—If you specified a directory other than the default, this directory was created in your specified location, and all of the files from the \VFE6\iLibs directory were copied into it.
    d. *MetaData*—This is the directory in which VFE will create your metadata. See Chapter 5, “Metadata,” for more information.
    e. *Misc*—A copy of the “default menu” is placed into this directory. You can read more about the default menu later in this chapter.
    f. *Output*—This is an empty directory in which you are expected to place your report layout files.
    g. *Progs*—This is an empty directory in which you are expected to place any PRG files you might create.
3. Perhaps most importantly, a subclass of VFE’s application class was created and placed in the class library file AAPP.VCX, which was placed in the Libs directory of your project. The application class is the class that is instantiated to run your application. This class has properties that are populated based on

information you entered in the wizard, such as company name, icon files and so forth.

Application Builder

The Application Builder is the main component of the VFE IDE, and it's where you will spend most of your time (see **Figure 2**). The builder is actually a replacement for the VFP Project Manager. You will see later that the VFP project is actually open but hidden, allowing VFE to reference and update the project using a project hook class. The Application Builder contains three pages, labeled Project, Classes and Files.


Visual FoxExpress - Application Builder: BOOKTUTORIAL.PJX

Application Name: Book Tutorial
Organization Name: F1 Technologies
Version: 1.0
Default Form Toolbar: DefaultToolBar

***Figure 2**. The Application Builder is command central while you are building your app.*

The Classes page lists all of the classes you have created for your application. The classes are organized in a tree view that is similar to the Classes page of the VFP Project Manager. However, instead of being listed by class library file, VFE categorizes the classes based on the primary building block classes of the VFE framework, which are the cursor, data item, data environment, business object, presentation object, view parameter container, form, toolbar, output and other. The application object is found in the "other" object category.

We will discuss each of these classes as we work through the framework and build a small tutorial application.

Project page

The Project page lists general information about your project (see Figure 2). This page of the Application Builder is more informative than it is productive. It contains information such as the name of your application, version number and so forth. It is from this page that you also access the DBCX Explorer and the Security setup. We will discuss both of these in detail later in the book.

Classes page

The Classes page provides a categorized view of the classes you have built for your application (see **Figure 3**). You will spend most of your time in the Application Builder on the Classes page. This page is similar to the Files page; however, instead of grouping the files in your project by file name and directory, they are grouped by class.

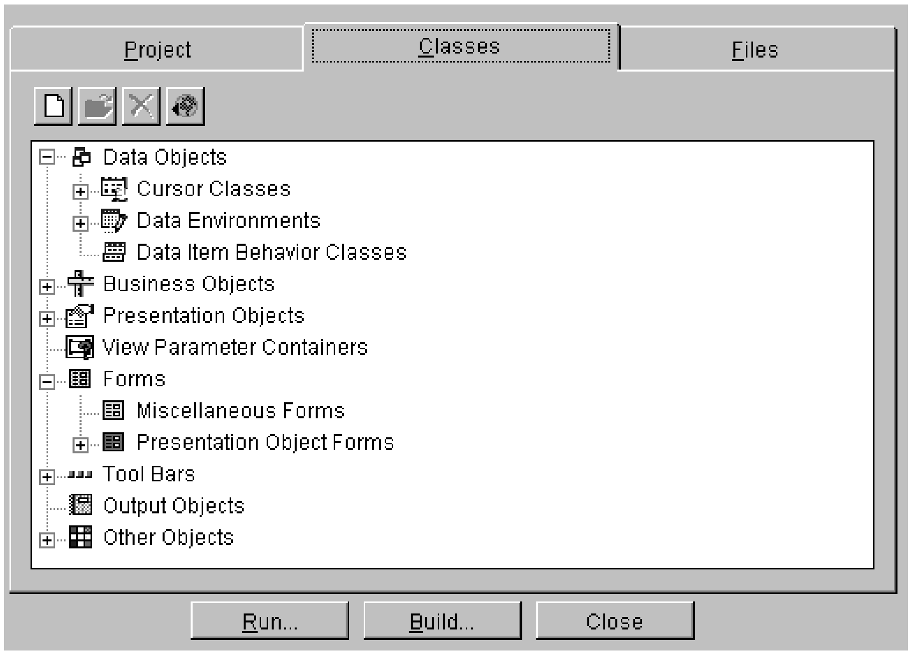


***Figure 3**. The Classes page categorizes the classes you've built.*

There is a slight similarity between the Classes page of VFE's Application Builder and the VFP Project Manager's Classes page, as both list all the class libraries in your project. But VFE takes the organization of your project a step further by classifying the classes list by category. Each of the major nodes on the Application Builder's Classes page tree view corresponds to one of the major class types of VFE. There are five major class types included in the VFE framework that you will work with: the cursor object, data environment object, business object, presentation object and form object.

Files page

The Files page, shown in **Figure 4**, allows you to access all of the items in your project without leaving VFE; you can even open the framework class libraries from here. Here you'll see a tree view that contains all the files that are in the VFP project. As a matter of fact, this page is very similar to the All page of the VFP Project Manager.

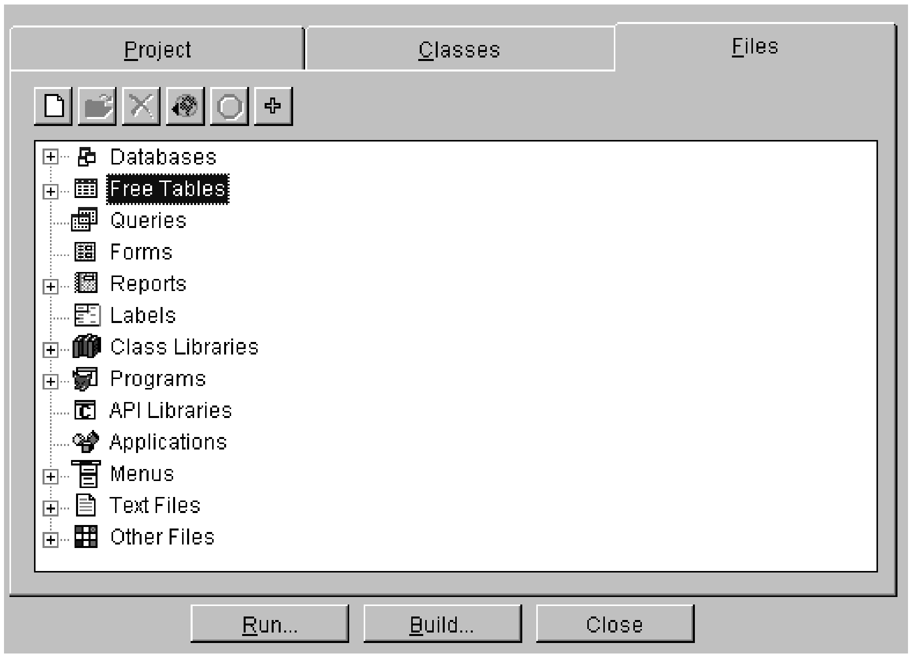


***Figure 4**. The Files page provides access to all files in your project, including the framework files.*

If you need to access a specific file in your application, this page will give you access to it. One thing to remember is that this is the only page from which you can access your menu files. If you've worked with previous versions of VFE, you know that this is a huge improvement. The Files page allows you to access all of the items.

Wizards

If you take a look at the Classes page of the Application Builder (shown in Figure 3), you will see that there are many classes you will be building. VFE has provided a wizard to create each of these classes for you. The wizards bring you through a series of steps requesting the information needed to build the class you are working on. Your responses will be used to populate properties in the class and also to add other classes to your class.

For example, a presentation object is a container for the interface controls you need in order to edit your business object data. The wizard will present you with a list of fields in the business object. For each field you select, an interface control class will be added to a presentation object.

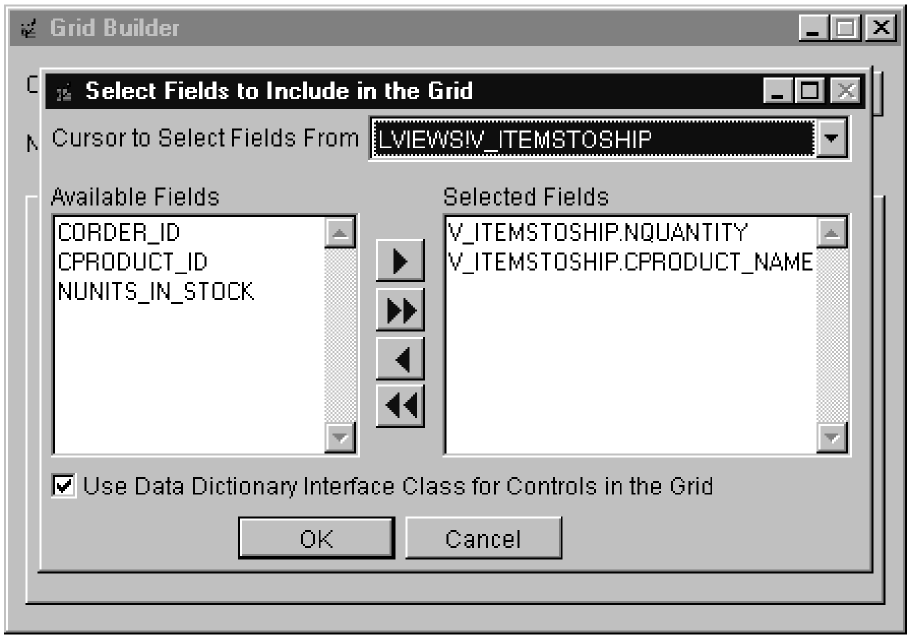
To run a wizard, you select the class type you want to build by highlighting a node on the tree view of the Classes page. Once the class type is highlighted, pressing the New button will bring up a wizard selection dialog. This dialog lists the wizards available for the class you selected. You see, you may want to build more than one type of form. You'll decide between the Modal Form wizard and the Business Object Form wizard.

One thing you should keep in mind is that the wizards are just tools to get you started. You don't *have* to use the wizards, and, by the same token, you are not limited to what the wizards build and how they build it. We will discuss this further in Chapter 16, "Extending the Framework."

Builders

Once you have created the classes with the wizards, you may want to edit them to your specific needs, or change them from how you originally specified them. Or you may want to modify the properties the wizard set when it built a class. Each class in the framework includes a builder. A builder is a small form/program that is written to set the properties and methods of a specific class.

You would launch a builder from the Class Designer. A builder provides access to commonly used properties and also may contain tools to simplify a certain task you need to perform to set up a control. The Grid Builder, for example, allows you to specify the columns you would like to be included in the grid (see **Figure 5**).


Grid Builder
Select Fields to Include in the Grid
Cursor to Select Fields From
Available Fields
Selected Fields
OK
Cancel

Cursor to Select Fields From: LVIEWS!V_ITEMSTOSHIP

Available Fields:
CORDER_ID
CPRODUCT_ID
NUNITS_IN_STOCK

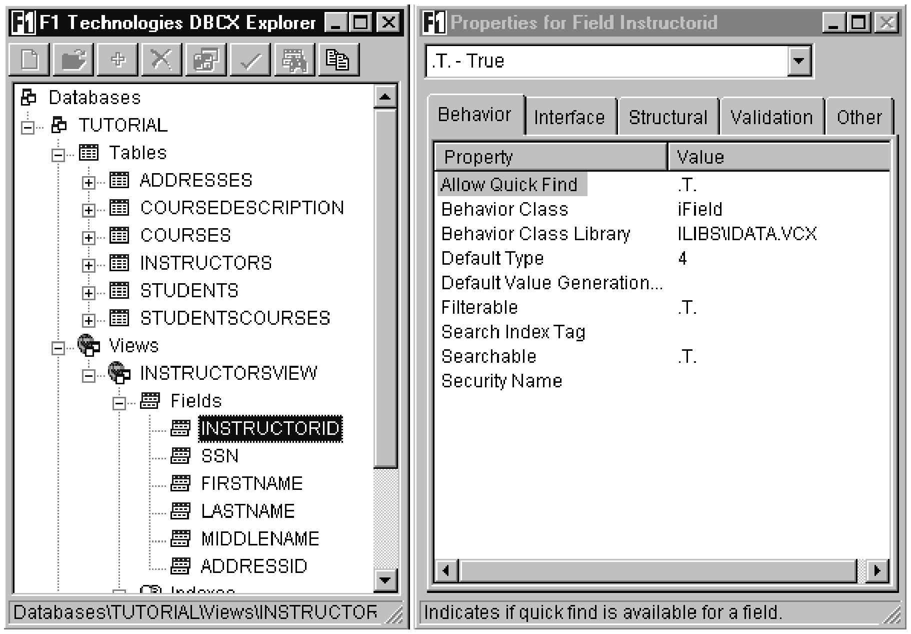
Selected Fields:
V_ITEMSTOSHIP.NQUANTITY
V_ITEMSTOSHIP.CPRODUCT_NAME

[✓] Use Data Dictionary Interface Class for Controls in the Grid

***Figure 5**. The Grid Builder contains a dialog that allows you to select and order the fields that will be used in the columns.*

DBCX Explorer

An important part of the VFE framework is the metadata. Metadata is information stored about the data items in the application database. Both the IDE and the framework use this data at both design time and run time. The DBCX Explorer is a tool provided in VFE, which you will use to edit the metadata (see **Figure 6**).


F1 Technologies DBCX Explorer
Properties for Field Instructorid
Behavior
Interface
Structural
Validation
Other

```
Databases
  TUTORIAL
    Tables
      ADDRESSES
      COURSEDESCRIPTION
      COURSES
      INSTRUCTORS
      STUDENTS
      STUDENTSCOURSES
    Views
      INSTRUCTORSVIEW
        Fields
          INSTRUCTORID
          SSN
          FIRSTNAME
          LASTNAME
          MIDDLENAME
          ADDRESSID
        Indexes
Databases\TUTORIAL\Views\INSTRUCTOR
```

.T. - True

Property	Value
Allow Quick Find	.T.
Behavior Class	iField
Behavior Class Library	ILIBS\IDATA.VCX
Default Type	4
Default Value Generation...	
Filterable	.T.
Search Index Tag	
Searchable	.T.
Security Name	

Indicates if quick find is available for a field.

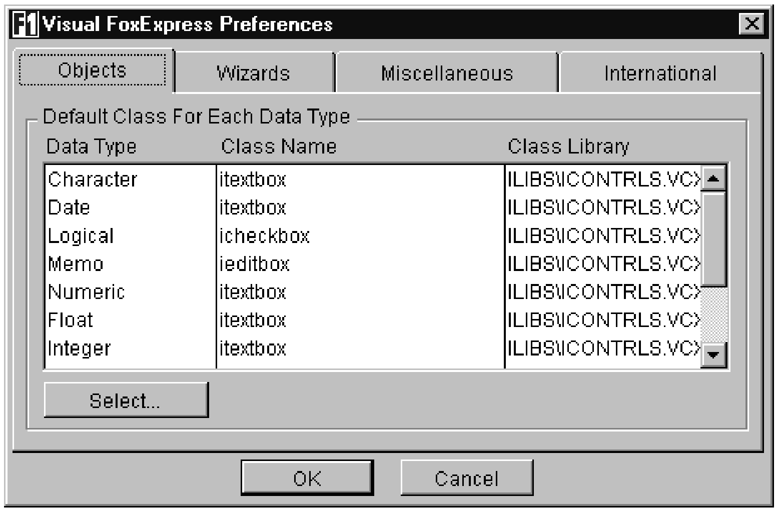
***Figure 6**. The DBCX Explorer allows the developer to edit metadata.*

The DBCX Explorer is a stand-alone set of windows that you will use alongside the Application Builder. The left window of this tool displays the data items that have been created for the application. The window on the right displays the extended attributes that have been created. Each attribute holds information about the highlighted data item.

Preferences

After installing VFE and running the executable either with a startup program or directly from the command line, you should have an Express menu pad on your VFP menu bar. The menu gives you access to the Application Manager, VFE's Help and tutorials, and a Tools menu. What we'll concentrate on here is the Preferences menu item.

If you open the Preferences dialog (see **Figure 7**), the first page you'll see is the Objects page. This is the page you'll wish you knew about before you started building your application if you find that you need to make changes to it. This page lists the particular VFE control class that will be designated when a field is added to the metadata. For example, if you add a text field to a presentation object, the control class for that text box will be set up as "iTextBox." This provides the same functionality for the VFE wizards that the Field Mapping page of the VFP Options dialog provides when you are in the Form Designer.



Visual FoxExpress Preferences

Objects | Wizards | Miscellaneous | International

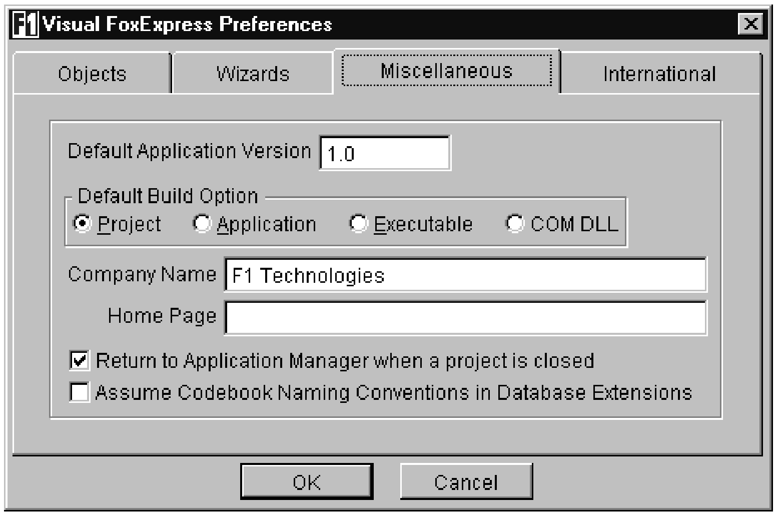
Default Class For Each Data Type

Data Type	Class Name	Class Library
Character	itextbox	\LIBS\ICONTRLS.VC
Date	itextbox	\LIBS\ICONTRLS.VC
Logical	icheckbox	\LIBS\ICONTRLS.VC
Memo	ieditbox	\LIBS\ICONTRLS.VC
Numeric	itextbox	\LIBS\ICONTRLS.VC
Float	itextbox	\LIBS\ICONTRLS.VC
Integer	itextbox	\LIBS\ICONTRLS.VC

Select... OK Cancel

***Figure 7**. The Objects page of the VFE Preferences dialog displays the class that will be assigned to fields with the matching data type.*

The Wizards page is pretty self-explanatory, so we'll skip to the Miscellaneous page (see **Figure 8**). You should be sure to visit this page prior to creating any VFE applications. The main check box you want to look at is "Assume Codebook Naming Conventions." When this is selected and you add a table to the metadata, VFE will chop off the first letter of each field name to populate the caption. We tend not to use Hungarian notation on field names, but if you do, you will always want to turn this check box on.



Visual FoxExpress Preferences

Objects | Wizards | Miscellaneous | International

Default Application Version: 1.0

Default Build Option: [✓ Project] Application Executable COM DLL

Company Name: F1 Technologies

Home Page:

[✓] Return to Application Manager when a project is closed

[ ] Assume Codebook Naming Conventions in Database Extensions

OK Cancel

***Figure 8**. The Miscellaneous page of the VFE Preferences dialog has the all-important "Assume Codebook Naming Conventions" check box.*

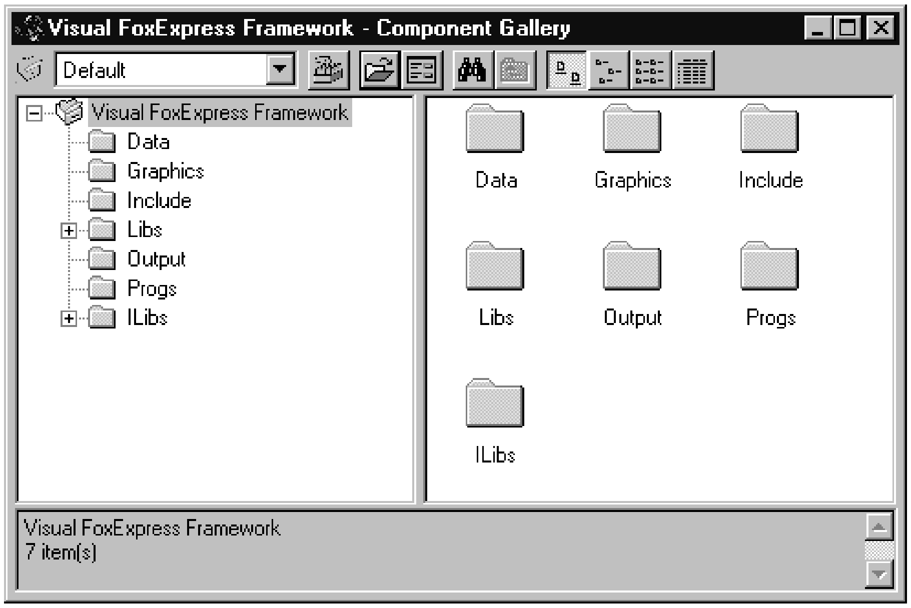
Express Tools

While these tools are the more common items with which you will become acquainted, there are a few more items of the IDE that we should take a look at. These items are loaded from the Express Tools menu pad of the Express menu.

Component Gallery

A feature that was added to Visual FoxPro, which was highly regarded by the programming community, was the Component Gallery. This program allows users to set up catalogs of files, grouping and organizing them as they see fit.

VFE has included a Component Gallery catalog that you can access for help when building your application. The gallery is fully functional and can be used when editing classes or writing code (see **Figure 9**).

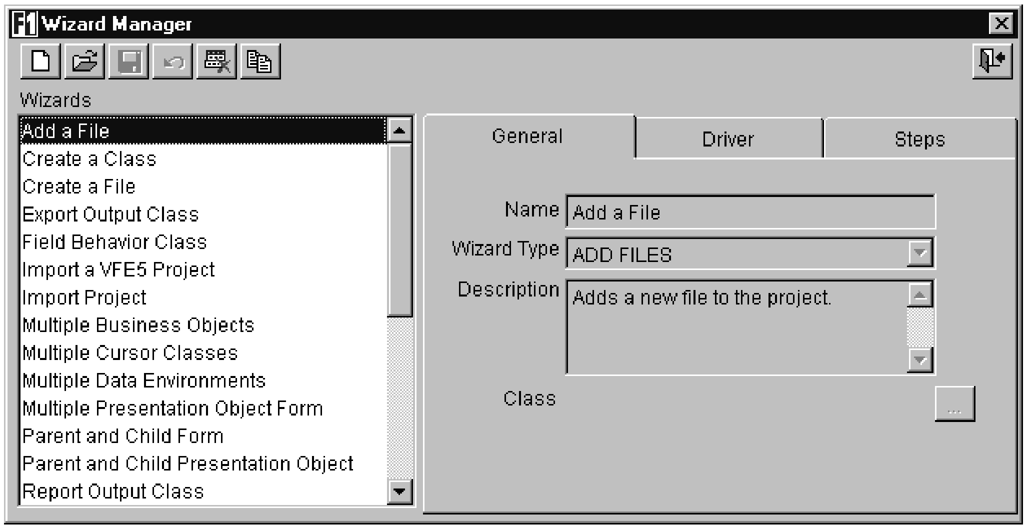


Visual FoxExpress Framework
7 item(s)

***Figure 9**. The Component Gallery provides quick and organized access to the framework classes.*

Wizard Manager

The Wizard Manager is a tool that allows you to modify the definitions and behavior of the systems wizards. As we mentioned earlier, the wizards are data-driven; each step is defined in a database. The Wizard Manager can modify this database. With the ability to modify the wizards, the framework can be more seamlessly extended by outside programmers (see **Figure 10**).



Add a File
Create a Class
Create a File
Export Output Class
Field Behavior Class
Import a VFE5 Project
Import Project
Multiple Business Objects
Multiple Cursor Classes
Multiple Data Environments
Multiple Presentation Object Form
Parent and Child Form
Parent and Child Presentation Object
Report Output Class

Name: Add a File
Wizard Type: ADD FILES
Description: Adds a new file to the project.
Class

***Figure 10**. The Wizard Manager allows you to edit the data that drives VFE's wizards.*

Edit Default Menu choices

Remember that when you created your first VFE application earlier, the default menu file was copied into the Misc directory of the project. Selecting "Edit Default Menu" from the Tools menu allows you to edit that default menu and customize it to your liking. Keep in mind that this is used for the default menu of all applications. The default is used if you don't change it; you are more than likely to change an application's menu once it has been created.

This function uses the standard VFP menu editor. The file is saved in the framework structure and copied to your application directory by the Application Manager while it is building your new project.

Summary

It's easy to see how using VFE is going to save you time and accelerate your productivity. The Application Manager provides a list of your projects, with tools to build and maintain the projects. The Application Builder gives you an interface that separates your work into categories used in the creation of a VFE application. As you work through the framework, you will recognize each of these categories as the main building block layers of the class libraries.

Finally, you are provided with tools to customize and modify the VFE IDE and framework to perform the way you want it to. It is not the intent of the framework to be an application cookie cutter where you stamp out similar applications. The customization tools are there so that you can insert your style and personality into the applications you build.

Now that you have peered into the toolbox, it's time to look at the building materials you will be molding with these tools. As many of us discover when visiting a home improvement store, you might find something that you didn't know existed before you entered but then decide that you just can't live without it.