Part I

INTRODUCTION TO DEVELOPER/2000 FORMS

He who has begun has half done. Dare to be wise; begin!
—Horace

CHAPTER 1  Developing the First Form
CHAPTER 2  Running the First Form
CHAPTER 3  Enhancing the First Form
Chapter 1

DEVELOPING THE FIRST FORM

Look with favor upon a bold beginning.
—Virgil

♦ The Equipment Tracking System
♦ Creating the First Form
♦ The Object Navigator
♦ Creating a Data Block with the Data Block Wizard
♦ The Layout Wizard
♦ Adding a Detail Block
♦ Compiling the Module
♦ Summary
In this chapter, you will develop the first Developer Forms application to solve a real-life problem. Before beginning the development activities, you will analyze the structure of entities, attributes, and database objects involved in this application.

Next, you will create a data entry form for the application. In the process, important elements of the Form Builder, such as the Object Navigator, toolbar, and message bar, will be discussed. The chapter also introduces basic Forms development techniques such as creating base table blocks, using master/detail relationships, and saving and generating modules.

1.1 THE EQUIPMENT TRACKING SYSTEM

Print and Press is a small desktop publishing company in Suburbia that prints everything, from The Suburban Sentinel, to postcards, to wedding invitations. It has a dozen employees, including Mrs. White, the general manager, and Mr. Brown, the receptionist. They all use desktop computers that run different software applications. In addition, the print shop has three scanners, two photocopying machines, and several printers.

1.1.1 STATEMENT OF THE PROBLEM

Mrs. White has hired you to put in order her bills and receipts for hardware and software purchases. Mr. Brown currently stores them in a file folder each time a new item is purchased. The problem is that, periodically, someone has to go through all the receipts in order to figure out the status of the assets as well how much money was spent on hardware and on software. You assess the situation, interview some of the employees who need these data the most, and decide to create a small Oracle Developer Forms application to solve their problem.

Before going any further, you should come up with a name for your application. Just like a sound bite, this name should convey as much information about the system as possible. The core functionality of the application should come across in no more than three or four words. The use of acronyms is acceptable, but you should be very careful to choose one that is easy to pronounce and remember. So, call the new application the Equipment Tracking System, or ETS for short.

1.1.2 DEFINING THE DATABASE OBJECTS FOR ETS

The main objects in the ETS application are the entities Hardware Asset and Software Asset. The first entity represents information about any electronic equipment purchased and located in the premises of Print and Press, that is used in the daily activity of the employees. Examples of hardware assets would by a personal computer (PC), a monitor, a scanner, or a printer. The entity Software Asset stores
Developing the First Form

data about any software package that is installed and runs on a particular piece of hardware. Examples of software assets are Windows NT, Microsoft Office, and other programs installed on Mr. Brown’s PC. The relationship between the entities Hardware Asset and Software Asset is shown graphically in Figure 1.1. This diagram is created in the Oracle Designer’s Entity Relationship Diagrammer, although many other tools are available to create such diagrams.

Each block in this diagram represents an entity. The line connecting the entities represents the relationship between the instances each entity may contain. From left to right, it can be read as follows: “Each Hardware Asset may be a location for one or more Software Assets.” It can also be read right to left, and, in that case, it becomes: “Each Software Asset must be installed on one and only one Hardware Asset.”

Identifying the entities of an application and the relationships among them is called entity modeling or data modeling. It is one of the most important activities in the process of designing information systems because it identifies the main facts and data elements, which the systems will create and maintain. Entities contribute directly to the definition and properties of the principal database objects for these systems. In the case of ETS, the tables HW_ASSETS and SW_ASSETS correspond directly to the entities Hardware Asset and Software Asset. Graphically, the tables that are part of a database system are presented in data diagrams similar in layout and purpose to the entity relationship diagrams. Figure 1.2 shows the data diagram for ETS created in the Oracle Designer’s Design Editor.

Besides the main objects of ETS and the relationship between them, this diagram also shows the columns and constraints for each table. The most important property of a column is its data type. Depending on the kind of data the column stores, its data type can be one of the predefined alphanumeric, numeric, date, or binary types. Object-relational versions of the Oracle Server, starting with Oracle8, support columns based on user-defined object types as well.

Alphanumeric columns represent character string data. The alphanumeric data types supported by the Oracle Server database include CHAR for fixed-length strings, and VARCHAR2 for variable-length strings, which is also the most flexible and frequently used alphanumeric data type. Numeric attributes have, in general, NUMBER data type and represent virtually any fixed and floating point numeric data, with up to 38 digits of precision. Attributes of DATE data type represent dates.

![FIGURE 1.1 The entity relationship diagram for ETS.](image-url)
in a flexible format that includes century, year, month, day, hours, minutes, and seconds. If the attribute will represent binary data up to 2000 bytes long, then RAW data type is used. For larger binary objects (up to 2 GB) such as graphics, bitmaps, and images, Oracle7 uses the LONG RAW data type. Oracle8 provides a more complete support for such objects through the large object (LOB) types, including CLOB for character objects and BLOB for binary objects.

A primary key (PK) constraint is made up of one or more columns that uniquely identify an instance of the database table. For example, the column SERIAL_NUM uniquely identifies each record in the table HW_ASSETS. A foreign key (FK) constraint on a table contains one or more columns that point to the columns that make up the primary key of another table. The foreign key provides reference integrity that guarantees that the values in that key match the values in the primary key of another table. For example, the foreign key constraint SW_HW_FK shown in the ETS data diagram contains the column HW_SERIAL_NUM. In the definition of the constraint, this column is associated with the primary key SERIAL_NUM of the table HW_ASSETS.

Every foreign key represents a relation between records in two tables. More precisely, it describes the dependence or child/parent relationship between the table that owns the foreign key on the table pointed at by the key. In the ETS application, the foreign key SW_HW_FK represents the fact that a software item is dependent on a hardware unit on which the software is installed.

Analyzing properties of attributes offers a good opportunity to identify the rules that data items will obey in the application. For example, in the ETS application, it is reasonable to ask that data satisfy the following rules:

- Names of hardware and software assets must be specified, hence they should not be NULL.
Developing the First Form

- The purchase cost of any hardware or software asset should be a positive value not to exceed $100,000.
- All dollar values should be specified in dollars and cents.

Some of these requirements, such as the first one, will be enforced at the database level. The rest of them could be implemented there as well, but you will enforce them in the application you are about to develop. Properties that determine whether the column of a table is a primary key, a foreign key, or the data it represents must not be NULL, must be unique, or obey other rules such as the ones described above, are also known as constraints. Often the constraints are represented visually in data diagrams, as in the case of the primary key and NOT NULL constraints for the tables HW_ASSETS and SW_ASSETS. As you can see in Figure 1.2, the required columns in the diagram are preceded by the asterisk (*) to set them apart from the optional ones which are preceded by a small “o” (for optional). The icon (#) identifies the columns that make up the primary key constraint of each table. Figure 1.3 lists the columns of the table HW_ASSETS and their properties.

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>DATA TYPE</th>
<th>LENGTH</th>
<th>CONSTRAINTS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Number</td>
<td>CHAR</td>
<td>6</td>
<td>PK, fixed length</td>
<td>Internal tracking number</td>
</tr>
<tr>
<td>Name</td>
<td>VARCHAR2</td>
<td>30</td>
<td>Not NULL</td>
<td>The name of the asset</td>
</tr>
<tr>
<td>Employee</td>
<td>VARCHAR2</td>
<td>30</td>
<td></td>
<td>The employee responsible for this asset</td>
</tr>
<tr>
<td>Location</td>
<td>VARCHAR2</td>
<td>30</td>
<td></td>
<td>The location of the asset</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>VARCHAR2</td>
<td>30</td>
<td></td>
<td>The name of manufacturing company</td>
</tr>
<tr>
<td>Model</td>
<td>VARCHAR2</td>
<td>20</td>
<td></td>
<td>Model name and number</td>
</tr>
<tr>
<td>Basic Warranty</td>
<td>VARCHAR2</td>
<td>30</td>
<td></td>
<td>Manufacturer warranty on the asset</td>
</tr>
<tr>
<td>Extended Warranty</td>
<td>VARCHAR2</td>
<td>30</td>
<td></td>
<td>Additional warranty on the asset</td>
</tr>
<tr>
<td>Vendor</td>
<td>VARCHAR2</td>
<td>30</td>
<td></td>
<td>The name of vending company</td>
</tr>
<tr>
<td>Purchase Cost</td>
<td>NUMBER</td>
<td>8, 2</td>
<td>Between $0.00</td>
<td>Cost of purchase for the asset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>and $100,000.00</td>
<td></td>
</tr>
<tr>
<td>Purchase Date</td>
<td>DATE</td>
<td></td>
<td></td>
<td>Date when asset was purchased</td>
</tr>
<tr>
<td>CS Phone</td>
<td>VARCHAR2</td>
<td>10</td>
<td></td>
<td>Customer support phone number</td>
</tr>
<tr>
<td>Notes</td>
<td>VARCHAR2</td>
<td>255</td>
<td></td>
<td>A brief description of the asset</td>
</tr>
</tbody>
</table>

**FIGURE 1.3** Columns for table HW ASSETS and their properties.
1.1.3 CREATING THE DATABASE OBJECTS FOR ETS

In order to follow the development activities discussed in the rest of this chapter and in the following two chapters, you must create the ETS tables in your database and load sample data in them. The section Installation of ETS Database Objects in the Preface provides the necessary instructions to complete this task. Please follow these instructions before continuing to the next section.

1.2 CREATING THE FIRST FORM

After creating the database tables, you are ready to begin developing the form for the ETS application. First, create a separate directory in your hard disk to store it. The remainder of this chapter will assume that this directory is C:\ETS. Now, launch the Form Builder following these steps:

1. Click the Start button in the Windows task bar.
2. Select Programs | Oracle Developer | Form Builder.

At this point, the Form Builder is started. If this is the first time you launch this application or you have not changed its default settings, you will be presented with the dialog box Welcome to the Form Builder shown in Figure 1.5. This dialog is a helpful utility when you are new to the Form Builder environment. It allows you to quickly begin the design of a new form by selecting one of the four
options in the Designing radio group. From here you can also access the Form Builder online help content. All the tasks that can be jump-started from the Welcome to the Form Builder dialog box are accessible from within the Form Builder. Once you become familiar with the Form Builder environment, you will want to stop this dialog from appearing each time you start the Form Builder. To achieve this, simply deselect the check box Display at startup under the Oracle Developer icon. To continue with the creation of the form, select the option Build a new form manually and click OK.

1.2.1 NAMING THE FORM

When the Form Builder is initially launched, it creates a new form module to help you get started with the development activities. This module is represented by the node MODULE1 in the hierarchy tree of objects shown in the Object Navigator window. Typically, you want to replace this name with another one that better represents the purpose of the module. The following steps allow you to rename the newly created module:

1. Click the MODULE1 entry in the list, if it is not already selected.
2. Click the same node again. Notice that the text of the node is selected.
3. Type “ETS” as the name of the new module.
4. Press ENTER when done.
1.2.2 SAVING THE FORM

The following are the actions required to save the form in the C:\ETS directory:

1. Select File | Save from the menu. Because this is a new module, the Windows’ standard Save As dialog box appears.

FIGURE 1.6  The Form Builder after saving the ETS module.
Developing the First Form

2. Do not modify the proposed name of the file or its type. This way, the module will be saved under the file ETS.fmb.
3. In the list box that displays the folder where the new file will be saved, select the folder C:\ETS.
4. Click Save.

At this point, the Form Builder saves the form according to your specifications. Your desktop should look similar to Figure 1.6. Notice in this figure that the name of file you just saved is shown in the status bar as well as in the title bar of the Form Builder window.

1.2.3 CONNECTING TO THE DATABASE

In order to proceed with the development of the ETS application, you need to connect to the database schema in which you created the tables HW_ASSETS and SW_ASSETS. These are the steps to connect:

1. Select File | Connect . . . from the menu. The Connect dialog box appears (see Figure 1.7).
2. Enter User Name and Password. The user name should be the Oracle account used to create the ETS tables, for example FORMS_DEV.
3. Enter the Database Service Name, SID or database alias, as it may apply to your environment. This entry uniquely identifies the database instance where the ETS tables reside.
4. Click Connect.

When the connection is established, the little icon to the left of the node Database Objects shows the expansion indicator “+.” If you click it, you will see all the database users whose objects you can access. By further expanding the nodes in the hierarchy tree, you can browse the database objects as shown in Figure 1.8. In this figure, account FORMS_DEV owns several tables that you will encounter as you read the rest of this book. Among them, you can see the table HW_ASSETS,
which is expanded to show its columns. The data type of each column is shown in parentheses.

FIGURE 1.8  Viewing database objects in the Object Navigator.

1.3 THE OBJECT NAVIGATOR

The window you have been working on so far is the Form Builder’s Object Navigator. The Object Navigator is the heart of not only the Form Builder, but also all the other Oracle Developer components. From here you can display, access, and
edit just about every object in an application. Figure 1.9 represents the Object Navigator.

The most important part of the Object Navigator is the hierarchy tree of objects that make up the application. The entries in the list are grouped and organized based on the type of each Form Builder object. As you can see from Figure 1.9, a form module can have objects such as triggers, alerts, and data blocks. Do not allow names such as "LOVs" or "Canvases" discourage you. As cryptic as they may seem now, you will see that they are really easy to use and manipulate.

Entries in the hierarchy tree are also called nodes. Small icons precede the label of each node. These icons play a very important informational and navigational role. The "empty square" icon means that the object type is not instantiated. In other words, there are no objects of that type in the application yet. To
create one, double-click the icon. If the icon has the indicator “+” inside the square, then there is at least one object of that type in the application. To see all the objects of a particular object type, click the “+” icon or, as it is often said, expand the node. When the node is expanded, the indicator “+” is replaced by the indicator “−”. All the instances of that object type are listed beneath, indented to the right of the object type node. To hide them, click the “−” icon, or collapse the node.

Along the left border of the Object Navigator window is a vertical toolbar. The buttons listed here allow fast access to often-used commands such as Open, Save, Cut, Paste, Expand, and Collapse. The bitmaps incorporated in these buttons clearly describe their purpose. However, you do not have to memorize the meaning of all these icons. Developer/2000 Forms uses popup help to assist you identifying and selecting the right button. To display a brief description of an icon, place the mouse on the icon and do not move it for an instant. A little box will pop up on the side, displaying the icon’s name.

1.4 CREATING A DATA BLOCK WITH THE DATA BLOCK WIZARD

Ultimately, any Form Builder application is a collection of interface items and boilerplate objects that allow users to visually access and modify data from the database. The interface items may correspond to columns in database objects, or may represent derived or computed data. All items in a form are grouped in a higher level of abstract objects, based on their meaning and functionality. These objects are called blocks. Blocks are often associated with database objects, in which case they are called data blocks. Blocks that are not associated with any particular database object are called control blocks. In the ETS application you will create two blocks that will interface with the tables HW_ASSETS and SW_ASSETS.

The process of creating a data block in the Form Builder can be summarized in the following steps:

1. Select the database object upon which the new block will be based (normally a table or view).
2. Specify the data bound items that will be part of the block (normally related to the columns of the selected database object).
3. Define the GUI layout of the block.

As you will see in the following two sections, the Data Block Wizard helps you complete the first two of these steps. The Layout Wizard, covered in Section 1.5, helps you accomplish the last step. Both these wizards, and other wizards en-
countered in the Form Builder, follow the metaphor for accomplishing multi-step tasks introduced and popularized by the Microsoft Windows graphical user interface (GUI). A wizard is essentially a dialog box with multiple panes attached to it. Each pane allows you to accomplish a specific activity required for the task. You can move between activities by clicking the buttons Next or Back. When you have completed all the required activities and are ready to complete the task, click Finish. At any time, you can exit the wizard by clicking Cancel or request help specific to the current activity by clicking Help.

To invoke the wizard, select Tools | Data Block Wizard from the menu. By default, the first pane of the wizard is a welcome pane that summarizes the main purpose and functionality of the wizard. Deselect the check box at the center of the pane if you do not want to see this pane the next time you invoke the wizard.

### 1.4.1 DEFINING THE DATA BLOCK TYPE

Essentially, a data block may be based either on a database table or view, or on a set of PL/SQL procedures stored in the Oracle RDBMS Server. In the ETS application all your data blocks will be based on database tables. You can convey this to the Data Block Wizard by selecting the radio button Table or View on the first pane of the wizard, as shown in Figure 1.10. After selecting this radio button, click Next to proceed to the second step.

### 1.4.2 DEFINING THE DATA ITEMS OF THE DATA BLOCK

The new block will present data from the table HW_ASSETS. Follow these steps to let the Wizard know about this.

1. Click the Browse button to the right of the text item Table or view. The Tables dialog box is displayed (see Figure 1.11). The checked items in this dialog mean that the Wizard will display only those tables that are owned by the current user. Depending on the application, you may want to display other types of objects that you or other users own. Select the appropriate check boxes to accomplish this.
2. Select the table HW_ASSETS from the list of available tables.
3. Click OK to close the dialog box and return to the Data Block Wizard.

At this point, the name of the selected table is entered in the text item Table or view. Notice that the list control Available Columns is populated with all the columns of the HW_ASSETS table. In order to include an item in the new block for each of these columns, click the button with the label “>>.” The visual effect of this action is that all the columns will be transferred from the list box Available Columns to the list box Database Items on the right. At this point, the
1.5 THE LAYOUT WIZARD

The Data Block Wizard helps you define the database items that will be part of the new block. These items will ultimately be displayed to the users in the GUI interface. While from the database interaction perspective all items are grouped in blocks, from the user interface perspective items may be grouped in frames. Each frame can be associated with only one block and a block may not have more than one frame.

The Layout Wizard is used to adjust the visible interface and GUI properties of items in a frame. Since they work hand-in-hand to create a new block, the Form Builder invokes the Layout Wizard as soon as the Data Block Wizard completes its task. In general, you can invoke the Layout Wizard independently by...
Developing the First Form

selecting Tools | Layout Wizard from the menu. As with the other wizard, deselect the check box in the Welcome pane if you do not want to see this pane the next time you invoke the wizard.

In Forms Builder applications, all the interface items are placed on objects of a special type called canvases. A canvas is then attached to a window frame and ultimately displayed to the user when the application is run. The first pane in the Layout Wizard allows you to specify the canvas object for the data block items and its type. Since there are no canvases previously created in the module, the only available option in the list box Canvas is (New Canvas), as shown in Figure 1.13. With the settings shown in this figure, the Layout Wizard will place the items of the new block on a new content canvas.

FIGURE 1.11  The Tables dialog box.
The next pane helps you define which items from the block HW_ASSETS will be displayed to the users. The list control Available Items in this pane is populated with all the items of block. In order to display all the items, click the button with the label “>>.” This will cause all the items to be transferred to the Displayed Items list box.

The next pane in the Layout Wizard helps you define the prompt that will accompany each displayed item, as well as the width and height. Based on the properties of the database column, such as the name and length, the wizard proposes settings for these properties. You can override these settings if necessary. As an example, you can change the prompt of the item SERIAL_NUM replacing the default setting “Serial Num” with “Serial Number,” and reduce the Width of the item Notes to 200, as shown in Figure 1.14. Click Next when you are ready to continue to the next step.

Now specify layout settings for the frame. The items in the HW_ASSETS block will be displayed in form style, therefore keep the radio button Form selected and click Next. In the new pane, type “Hardware Assets” in the Frame

**FIGURE 1.12** The Table pane of the Data Block Wizard.
Developing the First Form

Title text item; leave the other properties unchanged. Figure 1.15 shows the state of the current pane.

At this point, you are ready to complete the layout design of the HW_ASSETS block. Click Finish to close Layout Wizard. The Form Builder creates the frame Hardware Assets and brings up the Layout Editor containing the results of actions taken by the wizard.

1.6 ADDING A DETAIL BLOCK

In this section you will create a block that will interface with data in the SW_ASSETS table. The ETS users are mainly interested to know which software products are installed and run on their hardware equipment. You want the application to reflect this. In other words, if Mr. Brown’s PC is displayed in the HW_ASSETS block, the SW_ASSETS block should list all the programs installed
there. If Mrs. White’s PC is displayed, the software items should change accordingly.

In Form Builder, this type of coordination is implemented by defining a master-detail relationship between the HW_ASSETS and SW_ASSETS blocks. The HW_ASSETS block will be the master block, because it will drive the information displayed in the other block. The SW_ASSETS block will be the detail one, because it will supply additional information about the master record currently displayed in the HW_ASSETS block.

The process of creating any data block is very similar to what you did in Sections 1.4 and 1.5. Therefore, this section will not provide detailed instructions on how to create the SW_ASSETS block or define its layout. Instead, here is a list of the main steps you can execute:

1. Invoke the Data Block Wizard and request to create a block based on a table or view.

![Figure 1.14](image) The Items pane of the Layout Wizard.
Developing the First Form

2. Select SW_ASSETS as the base table for the new block, and include all its columns as database items in the block. At this point, the Data Block Wizard presents a new pane that you did not see when you created the HW_ASSETS block. This pane allows you to create a master-detail relationship between the blocks.

3. Click the Create Relationship button. The ETS: Data Blocks dialog box appears.

4. Select HW_ASSETS from the list and click OK to close the dialog box. The wizard intelligently supplies the join condition, as shown in Figure 1.16, based on column, primary key and foreign key definitions for both tables.

5. Click Finish to complete work with the Data Block Wizard and invoke the Layout Wizard.

6. Accept the selection of the existing canvas as the one where the items of the new block will be created.

FIGURE 1.15  The Rows pane of the Layout Wizard.

2. Select SW_ASSETS as the base table for the new block, and include all its columns as database items in the block. At this point, the Data Block Wizard presents a new pane that you did not see when you created the HW_ASSETS block. This pane allows you to create a master-detail relationship between the blocks.

3. Click the Create Relationship button. The ETS: Data Blocks dialog box appears.

4. Select HW_ASSETS from the list and click OK to close the dialog box. The wizard intelligently supplies the join condition, as shown in Figure 1.16, based on column, primary key and foreign key definitions for both tables.

5. Click Finish to complete work with the Data Block Wizard and invoke the Layout Wizard.

6. Accept the selection of the existing canvas as the one where the items of the new block will be created.
7. Select to display all the database items except for HW_SERIAL_NUM.
8. Choose a tabular layout of the frame for the SW_ASSETS block.
9. Set the Frame Title to “Software Asset”; set Records Displayed to “5” and select the check box Display Scrollbar.
10. Click Finish to complete the creation of the second block.

Once again, the Layout Wizard builds the layout of the specified block and displays the results in the Layout Editor. Save the work done so far by taking any of the following actions:

- Select File | Save from the menu.
- Press CTRL+S from the keyboard.
- Click the Save icon in the toolbar.
1.7 COMPILING THE MODULE

As in other programming environments, in order to run the newly created module, an executable file must be compiled first. During the compiling process, the Form Builder checks and validates all the objects in the module, their property settings, and the code attached to them. If conflicts are found, the appropriate error messages are generated and displayed to the user. The error messages are also written to a file that has the same name as the form module, and the extension “.ERR.” For example, if errors are encountered during the compilation of ETS.FMB module, they are written to the file ETS.ERR. The error file is created under the same directory as the binary file.

1.7.1 COMPILING FROM THE FORM BUILDER

There are two ways to generate a module that is open in the Form Builder:

- Select File | Administration | Compile File from the menu.
- Press CTRL+T from the keyboard.

If the compilation is successful, an executable file with the extension .FMX is created under the same directory as the binary file. For example, the executable file of the ETS.FMB module will be ETS.FMX.

1.7.2 SETTING FORM BUILDER PREFERENCES

In the Form Builder, like in other programming environments, the application development process repeats the following basic sequence of actions:

1. Change the application by adding to or modifying its functionality.
2. Create the new executable version by compiling.
3. Execute or run the application to assess the effects of additions and modifications.

Because it is so customary to generate the executable version of a module before running it, a setting in the Form Builder Preferences allows you to bypass Step 2 in the sequence above. When developing a form, you simply work with the application and then run it to see the effect of the modifications. The Form Builder automatically compiles the new executable before running it.

The setting for this and other options of the Form Builder can be accessed and modified in the Preferences dialog box. You can access this dialog by selecting Tools | Preferences . . . from the menu. Figure 1.17 shows the default settings for the General tab of this dialog box.
As you can see from this figure, if you set the Save before Building check box, you can skip the step of saving the form before generating and running it. Once an option is set, the setting will remain in effect until you change or reset it.

1.7.3 USING THE DEVELOPER FORM COMPILER

The methods described above compile a module from within the Form Builder, while you develop the application. There are instances when you may want only to compile a module, without doing any other design activities. These situations are mostly encountered when you upgrade from older versions of Oracle Developer or when you port your application from one platform to another. In these cases, you may use the Form Compiler tool to speed the process of creating new executables. The Form Compiler takes as input the binary files (.FMB) created in the Form Builder. If the generation process is successful, the executable (.FMX) is created. Follow these steps to invoke and use the Form Compiler:

1. Click the Start button in the Windows task bar.
2. Select Programs | Oracle Developer | Form Compiler. The Form Compiler Options window appears (see Figure 1.18).
3. Enter the name of the form module you want to compile, as shown in Figure 1.18. Click Browse to select the appropriate module if necessary.
4. Fill in the Userid, Password, and Database text items according to your environment.
5. Click OK.

If you do not supply the full name and location of the module, Forms Generate will search only the BIN folder of the Oracle Home folder of your environment. For example, if this folder is C:\ORACLE, then the directory searched by the compiler default is C:\ORACLE\BIN.

### 1.8 SUMMARY

This chapter introduced you to a simple application and guided you through the process of creating the first Form Builder module. Some of the most important concepts discussed here were:
The following table describes the software assets that were discussed in this chapter. From the main HTML page of the software utilities provided with the book follow the links Software and Chapter 1 to access these assets:

<table>
<thead>
<tr>
<th>ASSET NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH01.FMB</td>
<td>The ETS module created in this chapter.</td>
</tr>
</tbody>
</table>