AppForge MobileVB™ Tutorial

October 29, 2004

Contents

List of Tables

List of Figures

Overview

AppForge MobileVB™ is an extension of Microsoft® Visual Basic® that makes writing applications for mobile devices easier than ever.

Visual Basic is a powerful programming language that's easy to learn. You create the user interface by "drawing" controls, such as text boxes and command buttons, on a form. Next, you set properties for the controls to specify values such as caption, color, and size. Finally, you write code to bring the application to life. MobileVB works the same way, and includes many of the same functions and methods as Visual Basic.

This tutorial will take you through the steps of designing and writing code for a sample application using MobileVB™. You'll learn basic Ingot™ manipulation at design time and run time, programming for events, and database usage techniques.

If you've never used Visual Basic® before, we recommend reading the documentation for beginning programmers in the Microsoft® MSDN Library® at http://msdn.microsoft.com/library before attempting this tutorial.

Be sure to check out the section "MobileVB™ Support for Visual Basic" for details on functions, structures and methods supported by MobileVB.

TVToday Tutorial Application

The TVToday tutorial application provides TV program information for multiple channels in four categories for an entire day. The user can view all the programs that correspond to a given category within any one-hour range. TVToday also supplies detailed information about any TV program and a preview of the program, if available.

The TVToday application uses a range of AppForge Ingots™. Some Ingots are similar to Visual Basic controls, while others are unique to MobileVB.

Requirements

The TVToday Tutorial requires MobileVB™, which can be installed on any PC running Windows 95, Windows 98, Windows NT 4.0, Windows 2000, or Windows XP. If you are running Windows 95, you must install Windows 95 Service Pack 1. If you are running Windows NT 4.0, you must install Windows NT 4.0 Service Pack 4 or higher.

The Palm™ HotSync Manager® is required for uploading MobileVB applications to a Palm OS® device, and Microsoft® ActiveSync® is required to deploy an application to a Pocket PC device. However, you can still debug and run MobileVB™ applications within the Visual Basic environment without a handheld device.

NOTE: Palm OS® version 3.1 or later is required to run MobileVB apps on a Palm OS handheld device.

File Locations

Once MobileVB™ has been fully installed, all of the essential files reside in the AppForge directory. The following table lists all of the files required by this tutorial.

Folder Location	Files
...AppForge\Toolkit\doc\VBTutorial\Database	Category.PDB
	Program.PDB
...AppForge\Toolkit\doc\VBTutorial\Graphics	TVT_ARBD.RGX

	TVT_ARBH.RGX
	TVT_ARBK.RGX
	TVT_ARFD.RGX
	TVT_ARFH.RGX
	TVT_ARFW.RGX
	TVT_ARRT.RGX
	TVT_CGBH.RGX
	TVT_CGBK.RGX
	TVT_LOGO.RGX
	TVT_PGBH.RGX
	TVT_PGBK.RGX
	TVT_TVBH.RGX
	TVT_TVBK.RGX
	Mov_00.RGX
	Mov_01.RGX
	Mov_02.RGX
	Mov_03.RGX
	Mov_04.RGX
	Mov_05.RGX
	Mov_06.RGX
	Mov_07.RGX
	Mov_08.RGX
	Mov_09.RGX
	Mov_10.RGX
	Mov_11.RGX
...AppForge\Fonts	Palm_05.CMF
	Palm_05B.CMF
	Palm_07.CMF

Table 1: File Locations

As an aid in completing this tutorial, different versions of TVToday are provided with the install. Each version represents a development stage of the tutorial. They are broken down by lesson in the ...AppForge\Toolkit\doc\VBTutorial folder of your MobileVB™ install. Each folder contains the TVToday application, as it should appear following the completion of a lesson.

Lesson Overview

This tutorial is broken down into 11 lessons. Each lesson provides step-by-step instructions that introduce important techniques and MobileVB™ features.

Lesson	Description	What is Covered

Lesson 1	Creating a MobileVB™ Project	Creating a MobileVB Project
	Adding AppForge Ingots™	
	Saving and Running the Project	
Lesson 2	Adding Database Connectivity	Adding the Category Database
	Keeping Track of Database Fields	
	Creating and Calling a Subroutine	
	Saving and Running the Project	
Lesson 3	Creating the Category Form	Adding a New Form
	Copying Ingots from One Form to Another	
	Adding New Ingots	
	Creating Navigation Between Forms	
	Saving and Running the Project	
Lesson 4	Adding Connectivity to the Program Database	Adding Ingots To Access and Present Program Data
	Keeping Track Of Database Fields	
	Loading Database Information	
	Creating and Calling a Subroutine	
	Saving and Running The Project	
Lesson 5	Displaying Programs By The Hour	Adding Time-Based Code
	Determining The Top Of The Hour	
	Creating and Calling a Subroutine	
	Saving and Running The Project	
Lesson 6	Providing Control Over Program Timeframe	Adding Ingots To Control The Timeframe
	Displaying and Retaining The Current Timeframe	
	Allowing Users To Control The Timeframe	
	Saving and Running The Project	
Lesson 7	Creating A Program Form	Adding A New Form
	To Copy Ingots From One Form To Another	
	Adding New Ingots To The Program Form	
	Creating Navigation Between Forms	
	Creating Navigation From Program Form To Main Form	
	Saving and Running The Project	
Lesson 8	Displaying Program Information	Adding Display Ingots To The Program Form
	Formatting a Time Range	
	Creating and Calling a Subroutine	
	Loading Database Information	
	Saving and Running The Project	
Lesson 9	Creating A Preview Form	Creating the Preview Form
	Copying Ingots to the Preview Form	
	Adding New Ingots To The Preview Form	

	Enabling and Disabling an AFButton	
	Creating Navigation Between Forms	
	Saving and Running The Project	
Lesson 10	Displaying Program Previews	Adding the AFFilmstrip Ingot
	Setting the Frames Property	
	Playing and Stopping the Preview	
	Saving and Running The Project	
Lesson 11	Uploading The MobileVB™ Project	Setting the Dependencies
	Uploading the Project	
	Viewing Messages In The Compiler	

Table 2: Lesson Overview

Lesson 1: Creating a MobileVB™ Project

Begin creating the TVToday application by choosing New Project from the File menu, then selecting MobileVB™ Project in the New Project dialog box. Choose "Palm OS" in the Design Target Selection window.

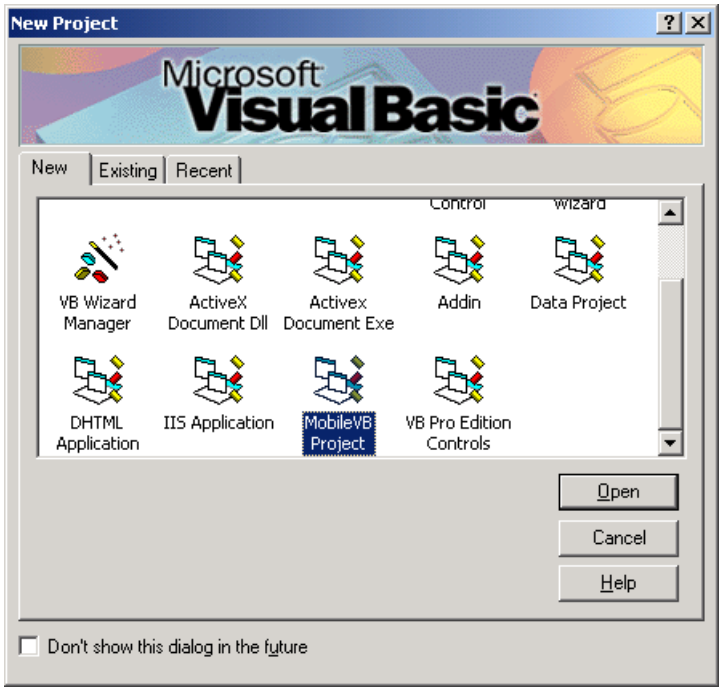

Figure 1: New Project Dialog

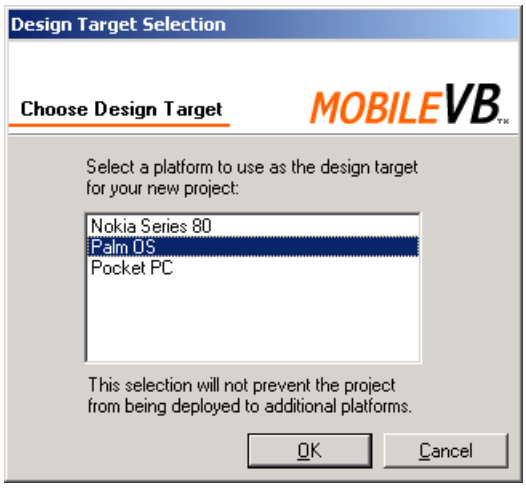

Figure 2: Design Target Selection

This will automatically create a blank MobileVB form.

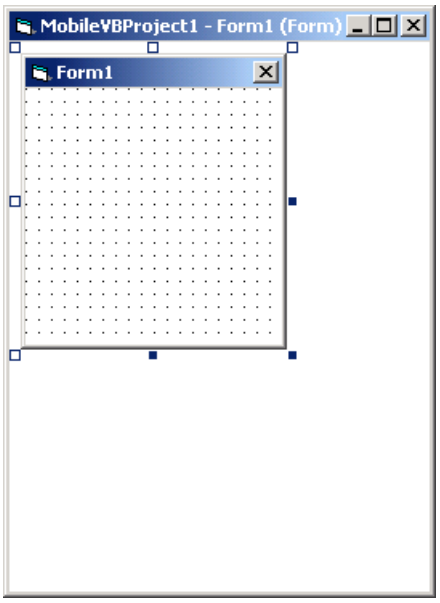

Figure 3: Blank MobileVB(tm) Form

You can change the names of projects and forms by clicking on them in the Project window, selecting Properties Window from the View menu, and changing the Name value. Change the Name of the Project from MobileVBProject1 to **TVToday**, change the name of Form1 to **frmTVTMain**, and clear the Caption property.

Next Select the **MobileVB Settings...** option from the MobileVB Menu.

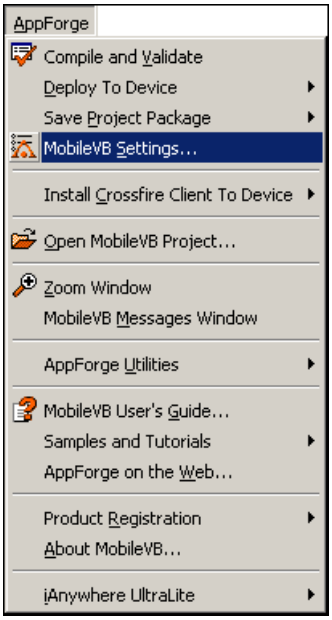

Figure 4: MobileVB(tm) Menu with MobileVB Settings Selected

This brings up the MobileVB Settings window.

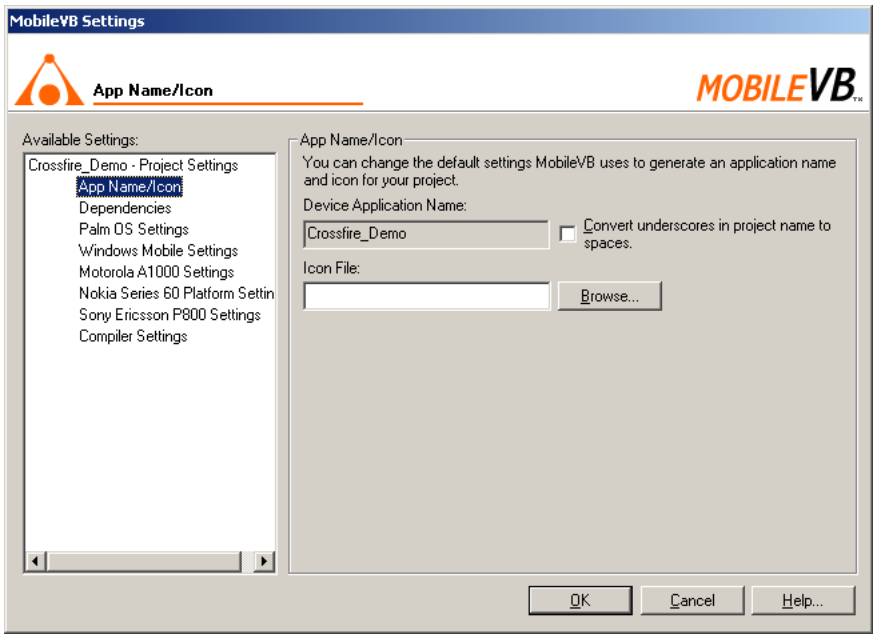

Figure 5: MobileVB(tm) Settings Window

Click on the **Palm OS Settings** tab. Make sure that the HotSync Name is set to the correct user. Change the CreatorID to AFTV (all capital letters). AFTV has already been registered with Palm, so click OK to continue.

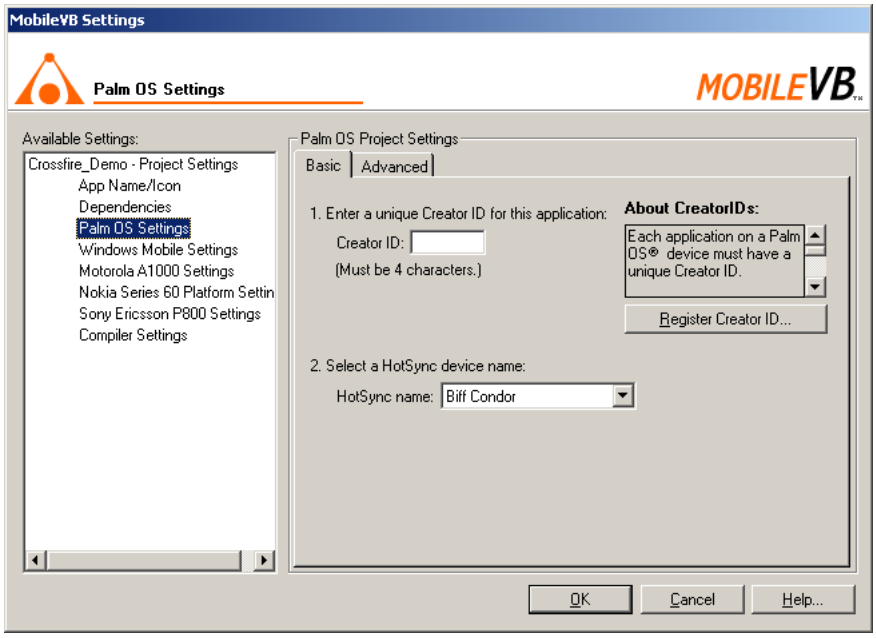

Figure 6: Palm OS Settings

Now save the project by taking the following steps.

1. Select **Save Project** from the File menu.

2. Save the following files with the corresponding file names (If the project and form names were set correctly, the file names should automatically default to the correct save names):

File	File Name
Form	frmTVTMain.frm
Project	TVToday.vbp

Table 3: Lesson 1 - Save Setttings

Adding Ingots to the Form

To create the look of the user interface for the TVToday Application, AppForge Ingots™ are added to the form. This first lesson of the tutorial uses the following Ingots:

Button	Name	Description
	AFShape	Rectangles or circles on your form.
	AFButton	Text button that can receive user events to begin, interrupt, or end a process.
	AFGraphic	Displays a graphic in the application. Valid graphic formats are .bmp, .rgx, .jpg, or .png*.
	AFListBox	Presents the user with a selectable list of text items.
	AFLabel	Functions as a text field that is not directly editable by the user.

Table 4: Lesson 1 - Ingot Table

** AppForge MobileVB 3.1 or later is required to utilize a .png file. See the Crossfire Client and PNG Files Section of the AppForge Crossfire Client And Additional Files documentation for details.*

First, add two AFShape Ingots to frmTVTMain. The AppForge Shape Ingot is similar to Visual Basic's own Shape control.

To create the first shape, select the AFShape Ingot in the Visual Basic toolbox and draw it anywhere on frmTVT-Main by clicking and dragging. In the Properties windows, set properties for the shape according to the following table. Keep all other properties at their defaults.

Ingot	Property	Setting	
	AFShape	Name	shpRect1
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	
	FillStyle	0 - Solid	
	Height	85	
	Left	0	
	Top	75	
	Width	38	

Table 5: Lesson 1 - shpRect1 Properties

Next, draw a second AFShape on the form. In the Properties windows, set properties for the second shape according to the following table. Keep all other properties at their defaults.

Ingot	Property	Setting	
	AFShape	Name	shpRect2
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	
	FillStyle	0 - Solid	
	Height	20	
	Left	38	
	Top	140	
	Width	35	

Table 6: Lesson 1 - shpRect2 Properties

If all of the settings are entered properly, frmTVTMain should look like the following figure.

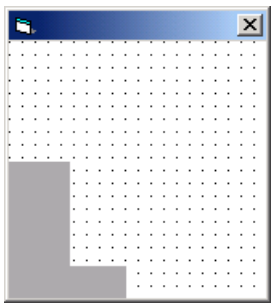

Figure 7: frmTVTMain with AFShape

Adding the TVToday Logo

The TVToday main form also features the TVT logo. To display this on frmTVTMain, add an AFGraphic Ingot to it.

Begin by selecting the AFGraphic Ingot in the Visual Basic toolbox and drawing it on frmTVTMain. In the Properties windows, set properties for the new AFGraphic according to the following table. Keep all other properties at their defaults.

Please Note: The source file for an AFGraphic must reside within the same folder or a sub-folder of the directory containing the project. When setting the Picture property of an AFGraphic or AFGraphicButton, a button labeled **"..."** appears. Click on this button to browse to the desired graphic. All graphics for the TVToday Application are stored in the ...AppForge\Toolkit\doc\VBTutorial\Graphics folder of your install of MobileVB™. Once the appropriate graphic has been selected, MobileVB allows you to copy the selected graphic into the folder containing the TVToday project.

As an alternative to using the browser window provided by MobileVB™, the graphic files for the project may be manually copied from the ...AppForge\Toolkit\doc\VBTutorial\Graphics folder to the folder in which you have saved the TVToday project. The Picture property may then be typed in manually. If the graphics are copied into a sub-folder within the TVToday project's folder, a relative path must be used for the Picture property. For example, if the graphic TVT_LOGO.RGX is copied into a folder named Graphics in the TVToday project's folder, the Picture property should be set to Graphics\TVT_LOGO.RGX.

Ingot	Property	Setting	
	AFGraphic	Name	gphLogo
	Height	75	
	Left	0	
	Picture	TVT_LOGO.RGX	
	Top	0	
	Width	75	

Table 7: Lesson 1 - gphLogo Properties

The form should now look like the following figure.

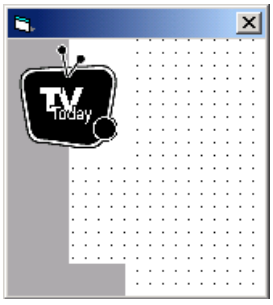

Figure 8: frmTVTMain with TVTLogo

Adding the AFButton, AFGraphic, and AFLabel Ingots

To complete the visual user interface of frmTVTMain, we need to add AFButton, AFGraphic, AFListBox, and AFLabel Ingots. Each Ingot is similar to a corresponding Visual Basic control.

Use the toolbox to draw an AFButton, AFGraphic, AFListBox, and AFLabel on frmTVTMain. In the Properties windows, set properties for the each Ingot according to the following tables. Keep all other properties at their defaults.

Ingot	Property	Setting	
	AFButton	Name	btnShowMe
	Appearance	1 - Generic 3D	
	BackColor	&H00AAAAAA&	
	Caption	show me	
	FontName	AFPalm	
	FontSize	12	
	FontStyle	1 (Bold)	
	Height	20	
	Left	74	
	Top	140	
	Width	71	
	AFGraphic	Name	gphArrowRight
	Height	20	
	Left	146	
	Picture	TVT_ARRT.RGX	
	Top	140	
	Width	14	
	AFListBox	Name	lstCatg
	FontName	AFPalm	
	FontSize	12	
	FontStyle	0	
	Height	111	
	Left	75	
	Top	28	
	Width	84	
	AFLabel	Name	lblCatg
	BackColor	&H00FFFFFF&	

Caption	Category:
FontName	AFPalm
FontSize	12
FontStyle	1 (bold)
Height	18
Left	75
Top	8
Width	73

Table 8: Lesson 1 - btnShowMe, gphArrowRight Properties

If all of the settings are entered properly, frmTVTMain should look like the following figure.

Figure 9: frmTVTMain with AFButton, AFGraphic, AFListBox, and AFLabel Ingots

Adding and Programming Sub Main

For all MobileVB™ applications, a Sub procedure in Main is required. Sub Main is the first procedure that is run when the application is started. We need to create a code module to contain Sub Main. Begin by selecting **Add Module** from the Project Menu.

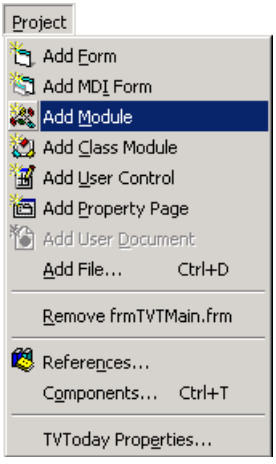

Figure 10: Add Module

This should bring up the Add Module Dialog box. Click on the **Open** button to add a new module to your project. Set the name property of the newly created module to mdlTVTMain.

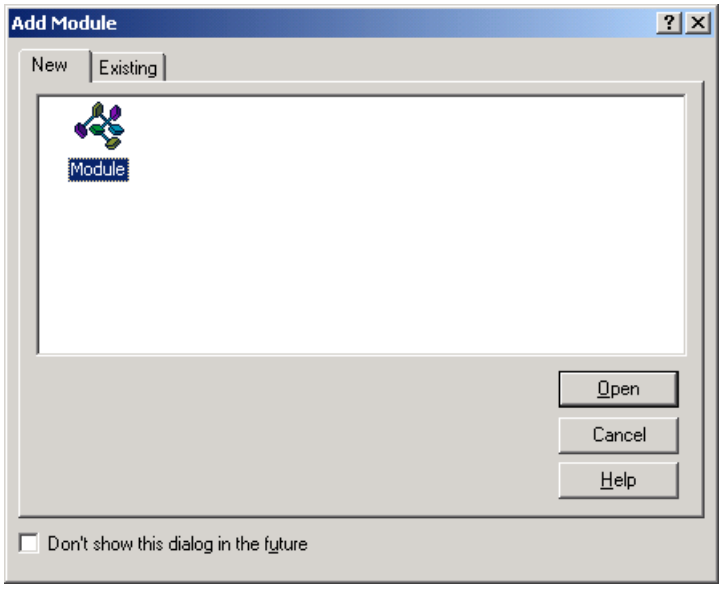

Figure 11: Add Module Dialog Box

The Sub Main procedure should reside within mdlTVTMain. Paste the following code into mdlTVTMain:

```
Sub Main()

  'show form
  frmTVTMain.Show

End Sub
```

This code shows frmTVTMain. In order to make TVToday start at Sub Main, it must be set as the startup object. This is done by selecting **TVToday Properties...** from the Project Menu. This brings up the TVToday Project Properties window. Set the Startup Object property in the upper right hand corner to Sub Main. Now TVToday will execute the code in Sub Main before any other code in the program. Click OK to return to the project.

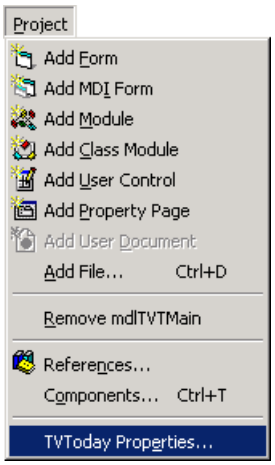

Figure 12: Selecting TVToday Properties... from the Project Menu

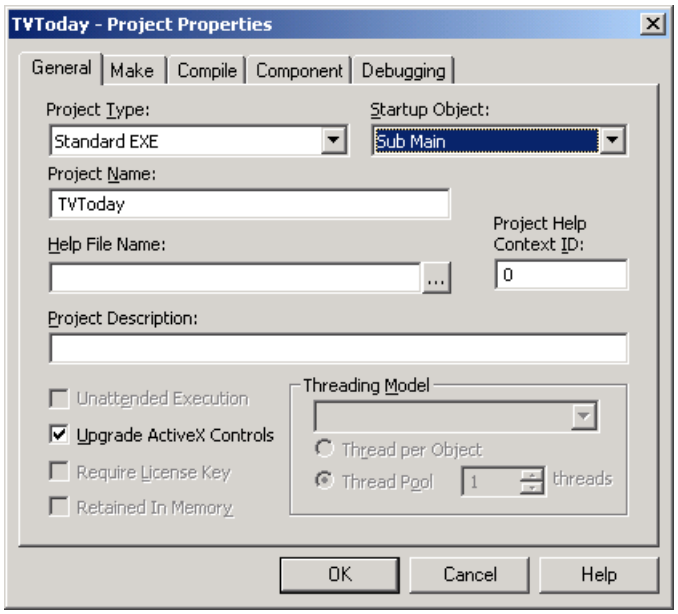

Figure 13: Setting the Startup Object for TVToday

Saving the Project

Before running the application for the first time, save the project.

To save the project

1. Select **Save Project** from the File menu.
2. Save the following files with the corresponding default file names:

File	File Name
Standard Module	mdlTVTMain.bas
Form	frmTVTMain.frm
Project	TVToday.VBP

Table 9: Lesson 1 - Save Setttings

Running the Project

To run the application, choose **Start** from the Run menu, or click the Start button on the toolbar. The TVToday Main form should appear. To exit, choose **Stop** from the Run menu, or click the Stop button on the toolbar.

Lesson 2: Adding Database Connectivity and Creating an Error Form

When the users enter the main form of the TVToday application, we want them to see a list of the available categories for today's TV programs. This list of available categories will come from a Palm Database (.PDB) and be displayed in an Appforge ListBox on frmTVTMain.

Figure 14: Main screen of the TVToday Application

To create the user interface for the Error form, we need to add AppForge Ingots to it. This lesson uses the following Ingots:

Button	Ingot	Description
	AFShape	Rectangles or circles on your form.
	AFButton	Text button that can receive user events to begin, interrupt, or end a process.
	AFGraphic	Displays a graphic in the application. Valid graphic formats are .bmp, .rgx, .jpg, or .png*.
	AFLabel	Functions as a text field that is not directly editable by the user.
	AFTextBox	Displays text on the form, or provides a text input box for the user.

Table 10: Lesson 2 - Ingot List

** AppForge MobileVB 3.1 or later is required to utilize a .png file. See the Crossfire Client and PNG Files Section of the AppForge Crossfire Client And Additional Files documentation for details.*

Copying the Category Database to the Project Folder

The category database for TVToday is installed as part of the typical installation of MobileVB™. In order to simplify access to the category database, copy it into the same folder as the TVToday application. The filename for

the category database is Category.PDB, and it is installed in the ...\AppForge\Toolkit\doc\VBTutorial\Database folder of your MobileVB install directory.

Keeping Track of Database Fields

To retrieve information from the database, the application code needs to target specific database records and specific fields. Each field is identified by an integer value.

The TVToday application uses constants to keep track of each database field. Add the following code into the Declarations section of mdlTVTMain:

```
'category database fields
Global Const LNG_CATG_CATGID As Long = 0
Global Const LNG_CATG_CATGNAME As Long = 1
```

Declaring the Category Database

MobileVB™ uses a Long type value to represent a Palm database. Since the category database is used throughout the TVToday program, we want to declare a public variable to represent it throughout our project. To do this, add the following code into the Declarations section of mdlTVTMain:

```
'category database
Public glngCatg As Long
```

Opening the Database

Next, the category database must be opened. The added code also checks to see whether the database was successfully opened. If the PDBOpen call was successful, frmTVTMain is shown. Code to load the categories as well as handle an unsuccessful PDBOpen call will be added in this lesson.

Edit the Sub Main procedure of mdlTVTMain so it matches the following code:

```
Sub Main()
  Dim strCatgPath As String

  'Use conditional compilation to determine if
  'program is running on Palm or Windows
  #If APPFORGE Then
    'Palm Path
    strCatgPath = "Category"
  #Else
    'Windows Path
    strCatgPath = App.Path & "\Category"
  #End If

  'open the database
  glngCatg = PDBOpen(Byfilename, strCatgPath, _
   0, 0, 0, 0, 0)

  'Check to see if database was successfully opened
```

```
  If glngCatg = 0 Then
    'Database was not successfully opened
    'Run the Load Error subroutine and
    'show frmTVTError
    frmTVTError.LoadError
    frmTVTError.Show
  Else
    'show form
    frmTVTMain.Show

  End If

End Sub
```

There are two important points to note about the path argument of the PDBOpen method:

1. The path argument for the PDBOpen method is different when running on a device and running in Windows®. Conditional compilation is used to ensure the proper path is assigned for each.
2. The database name is case-sensitive. Make sure that the name is typed in the correct case.

Loading the Categories

To load each category in the main form's AFListBox, a new Sub procedure called LoadCategories should be added to frmTVTMain. LoadCategories clears the AFListBox and loads each category from the database, record by record.

Paste the following code into the frmTVTMain code window:

```
Public Sub LoadCategories()
  'load the categories listed in database into the
  'listbox, tagging each listbox item with it's
  'corresponding UniqueID in the database

  Dim strCatgName As String

  'clear the listbox
  lstCatg.Clear

  'sort the database by category name
  PDBSetSortFields glngCatg, LNG_CATG_CATGNAME

  'start with the first record
  PDBMoveFirst glngCatg

  'while not on the last record of the database
  While Not (PDBEOF(glngCatg) = True)

    'get the record's category name
    PDBGetField glngCatg, LNG_CATG_CATGNAME, _
     strCatgName
```

```
      'add  the value as the next item in the listbox
      lstCatg.AddItem strCatgName, -1

      'select the item just added to the listbox
      lstCatg.ListIndex = (lstCatg.ListCount - 1)

      'set the list itemdata property as the unique ID
      'of the record
      lstCatg.ItemData(lstCatg.ListIndex) = _
       PDBRecordUniqueID(glngCatg)

      'go on to the next record in the database
      PDBMoveNext glngCatg
    Wend

    'select the first item in the listbox
    lstCatg.ListIndex = 0

End Sub
```

Note that as each record's category name is added as an AFListBox item, the code also tags the item with the record's unique ID. The unique ID is a long value that uniquely identifies a specific database record. We will use it later to find all the TV programs corresponding to a category.

Calling LoadCategories

Now the LoadCategories subroutine must be called so that the ListBox is filled before frmTVTMain is shown. Edit the end of Sub Main in the mdlTVTMain module so it matches the following (the bolded code indicates the newly added lines):

```
    Else

      'load the categories from the database into
      'the Main form's listbox
      frmTVTMain.LoadCategories

      'show form
      frmTVTMain.Show

    End If

End Sub
```

Adding and Enabling the Error Form

Earlier in this lesson, code was inserted into Sub Main to verify the successful opening of the Categories database. In this section we will add a new form and code used to notify the user that the database failed to open.

Adding a New Form to the MobileVB™ Project

1. Select **Add Form** from the Project menu. This will bring up the Add Form dialog box.
2. Select Form from the Add Form dialog box.
3. Click the Open button.

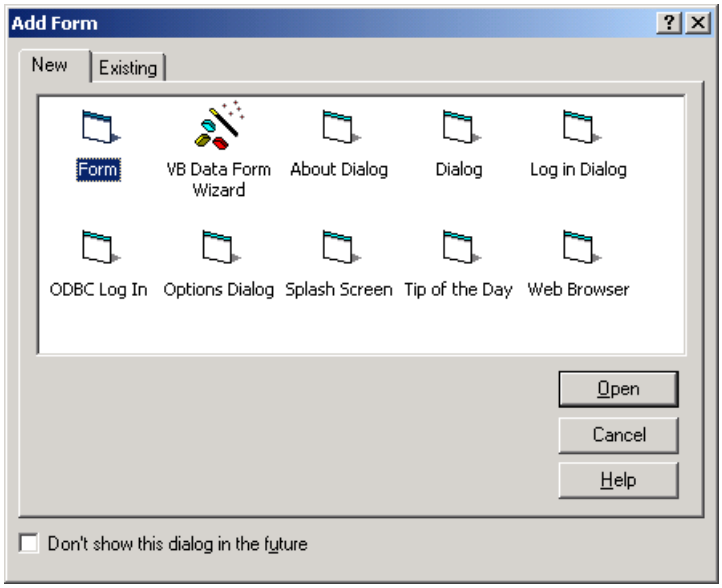

Figure 15: Add Form Dialog Box

In the Properties window, set the new form's Name property to **frmTVTError** and clear the Caption property.

Some of the Ingots we need for frmTVTError are nearly identical to those used on frmTVTMain. To save time, copy the matching Ingots from frmTVTMain to frmTVTError. The six matching Ingots are named gphLogo, shpRect1, shpRect2, btnShowMe, gphArrowRight, and lblCatg. Follow the following steps to copy these Ingots:

To copy Ingots from one form to another:

1. Open the main form, frmTVTMain. You can double-click on the form's name in the Project Explorer window to show it.
2. Click on the Edit menu and choose Select All to select all Ingots.
3. Copy the components by selecting Copy from the Edit menu, or by clicking the Copy button on the toolbar.
4. Open the Ingots' destination form, frmTVTError.
5. Paste the components by selecting Paste from the Edit menu, or by clicking the Paste button on the toolbar.

Once the Ingots are copied to the Error form, two of the Ingot name properties should be changed. On frmTVTError change the names as follows:

Ingot	Previous Name	New Name
	btnShowMe	btnExit
	lblCatg	lblError

Table 11: Lesson 2 - Copied Ingot Name Changes

Now make the following properties changes:

Ingot	Name	Property	Setting
	btnExit	Caption	exit
	lblError	Caption	Error:

Table 12: Lesson 2: Copied Ingot Changes

Adding the Error Message TextBox

The final Ingot placed on frmTVTError is an AFTextBox to display the error message. First, click on frmTVTError and delete the lstCatg list box. Then, use the toolbox to draw an AFTextBox on frmTVTError. In the Properties window, set properties for the Ingot according to the following tables. Leave all other settings at their defaults.

Ingot	Name	Property	Setting
	AFTextBox	Name	txtErrorMsg
		Alignment	0 - Left Justify
		BackColor	&H00FFFFFF&
		BorderStyle	1 - Single
		FontName	AFPalm
		FontSize	12
		FontStyle	0
		Height	113
		Left	72
		Multiline	True
		Text	Error Message
		Top	24
		UnderlineStyle	0 - None
		Width	88

Table 13: Lesson 2 - AFTextBox Property Settings

frmTVTError should now look like the following figure.

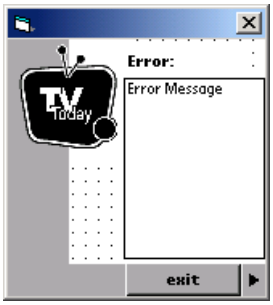

Figure 16: Error Screen of the TVToday Application

Adding Code to the Form

The final step in enabling the Error form is adding the code associated with it. First the LoadError Subroutine should be added. Click on frmTVTError, select Code from the View menu, and paste the following code in the window:

```
Public Sub LoadError()
  Dim strDbName As String

  'Identify the missing database
  If glngCatg = 0 Then
    strDbName = "Category.PDB"
```

```
    End If

    'Set the text for the Error Message text box
    txtErrorMsg.Text = strDbName & _
     " not found. Please HotSync " & strDbName _
     & " and run TVToday again."

End Sub
```

The above code identifies the missing database and displays the error message. Remember that LoadError is called before frmTVTError is shown.

If the user gets to this error form, it probably means the category database has not been loaded on the Palm OS® device. Therefore, the only option left to the user is to exit the program and load the database onto the device. To enable the user to exit the program from frmTVTError, double click on btnExit, and add the following code:

```
Private Sub btnExit_Click()
    'Ends the program
    End
End Sub
```

This completes the implementation of the Error Form.

Saving the Project

Save the changes made to the project by selecting Save Project from the File menu. Save the following file with the corresponding file name:

File	File Name
Form	frmTVTError.frm

Table 14: Lesson 3 - Save Names

Running the Project

Next, run the application by choosing **Start** from the Run menu, or by clicking the Start button on the toolbar. The main form will appear with a list of categories within the AFListBox. To test the Error form functionality, temporarily remove Category.PDB from its current location and run TVToday again. The frmTVTError form should appear instead of frmTVTMain. (Don't forget to return Category.PDB after testing the Error form's functionality.)

Lesson 3: Creating a Category Form

In this lesson, a new form is added to display TV programs within a chosen category. Once users have selected a category on the Main form, they can click the Show Me button to view the corresponding program on a second form. The Category Form looks like the following figure.

Figure 17: Category Screen of the TVToday Application

Lesson 3 of the tutorial uses the following Ingots:

Button	Ingot	Description
	AFShape	Creates rectangles or circles on your forms.
	AFButton	Text button that can receive user events to begin, interrupt, or end a process.
	AFGraphic	Displays a graphic in the application. Valid graphic formats are .bmp, .rgx, .jpg, or .png*.
	AFLabel	Functions as a text field that is not directly editable by the user.
	AFGraphicButton	Works like the standard button, but instead of a caption, it can specify different graphics for the up and down click positions.

Table 15: Lesson 3 - Ingot List

** AppForge MobileVB 3.1 or later is required to utilize a .png file. See the Crossfire Client and PNG Files Section of the AppForge Crossfire Client And Additional Files documentation for details.*

Adding a New Form

Before laying out the user interface for the Category form, a new form needs to be added to the project.

1. Select **Add Form** from the Project menu. This will bring up the Add Form dialog box.

2. Select Form from the Add Form dialog box.
3. Click the Open button.

In the Properties window, set the form's name property to **frmTVTCatg** and clear the Caption property.

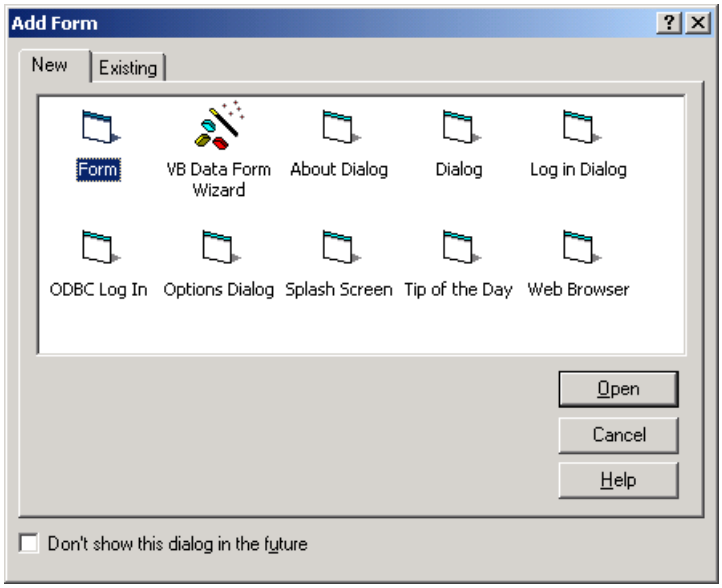

Figure 18: Add Form Dialog Box

Copying Ingots to the Category Form

Three of the Ingots we will use on the Category form are identical to Ingots on the Main form. To save time, copy the matching Ingots from the Main form to the Category form. The three matching Ingots are shpRect2, btnShowMe, and gphArrowRight.

To copy Ingots from one form to another:

1. Open the main form, frmTVTMain. You can double-click on the form's name in the Project Explorer window to show it.
2. Select the first component to copy by clicking on it in the form.
3. Select each additional component to copy by holding down the Shift key and clicking on each component you want.
4. Copy the components by selecting Copy from the Edit menu, or by clicking the Copy button on the toolbar.
5. Open the Ingots' destination form, frmTVTError.
6. Paste the components by selecting Paste from the Edit menu, or by clicking the Paste button on the toolbar.

When the AFShape, AFButton, and AFGraphic Ingots are pasted onto the Category form, they will be placed at the top of the form. Update each Ingot's properties to the following values:

Ingot	Name	Property	Setting
	shpRect2	Left	0
		Top	140
		Width	73
	btnShowMe	Left	74
		Top	140
	gphArrowRight	Left	146
		Top	140

Table 16: Lesson 3: Top Property Values

Adding Ingots to the Category Form

Use the toolbox to draw an AFLabel, an AFGraphicButton, and two AFShape Ingots on frmTVTCatg. In the Properties window, set properties for each Ingot according to the following table. Use the default setting for all other properties.

Ingot	Property	Setting	
	AFGraphicButton	Name	grbTVBack
	DownPicture	TVT_TVBH.RGX	
	FocusPicture	TVT_TVBK.RGX	
	Height	27	
	Left	0	
	NoFocusPicture	TVT_TVBK.RGX	
	Top	0	
	Width	40	
	AFShape (1)	Name	shpRect3
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	
	FillStyle	0 - Solid	
	Height	6	
	Left	40	
	Top	5	
	Width	120	

	AFShape (2)	Name	shpRect4
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	
	FillStyle	0 - Solid	
	Height	6	
	Left	40	
	Top	21	
	Width	120	
	AFLabel	Name	lblTVToday
	BackColor	&H00AAAAAA&	
	Caption	TVToday	
	FontName	AFPalm	
	FontSize	12	
	FontStyle	1 (Bold)	
	Height	12	
	Left	40	
	Top	11	
	Width	120	

Table 17: Lesson 3 - New Ingot Properties

The form should now look like the following figure.

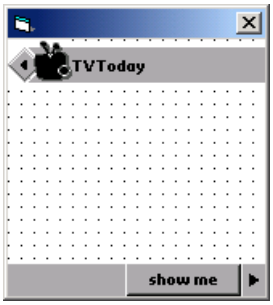

Figure 19: frmTVTCatg with AFLabel, AFShapes, and AFGraphicButton

Creating Navigation between Forms

When users click the Show Me button on the Main form, the Category form should replace the Main form. Double-click the Show Me button on the Main form to view the button's Click event in the Code window. Paste the following code into frmTVTMain's form module:

```
Private Sub btnShowMe_Click()
  'hide this form and show the category form
  frmTVTMain.Hide
  frmTVTCatg.Show
End Sub
```

To complete the navigation between forms, we need to add code to frmTVTCatg. When users click on the AFGraphicButton at the top of the Category form, the Main form should replace the Category form. Double-click the AFGraphicButton on the Category form to view grbTVBack's Click event in the Code window.

Paste the following code into frmTVTCatg's form module:

```
Private Sub grbTVBack_Click()
  'hide this form and show the main form
  frmTVTCatg.Hide
  frmTVTMain.Show
End Sub
```

Saving the Project

Save the changes made to the project by selecting Save Project from the File menu. Save the following file with the corresponding file name:

File	File Name
Form	frmTVTCatg.frm

Table 18: Lesson 3 - Save Names

Running the Project

Run the application by choosing **Start** from the Run menu, or by clicking the Start button on the toolbar. The main form will appear with a list of categories within the AFListBox. Click on the Show Me button to navigate to the Category form. Once on the Category form, click on the AFGraphicButton at the top left to return to the Main form.

Lesson 4: Adding Connectivity to the Program Database

When users choose a category and navigate to the Category form, they should see a list of television programs corresponding to the chosen category.

Figure 20: Category Screen of the TVToday Application

The data for these television programs will come from the Program database, Program.PDB.

Copying the Program Database to the Project Folder

The program database is installed as part of the typical install of MobileVB™. In order to simplify access to the program database, copy it into the same folder as the TVToday application. The filename for the program database is Program.PDB, and it is installed in the ...\Toolkit\doc\VBTutorial\Database folder of your MobileVB install directory.

Adding Ingots to the Category Form

To present the program information, the following Ingots will be added during Lesson 4:

Button	Ingot	Description
	AFShape	Used to create rectangles or circles on your forms.
	AFGrid	Used to display text or graphics in a grid. It also allows the user to select rows or cells in the grid. The grid can contain a title, fixed rows and columns, and has flexible options for colors, alignment, and borders.
	AFLabel	Functions as a text field that is not directly editable by the user.

Table 19: Lesson 4 - Ingots Added

Use the toolbox to draw an AFShape, AFGrid, and AFLabel on frmTVTCatg. In the Properties window, set properties for the each Ingot according to the following table. Use the default setting for all other properties.

Ingot	Property	Setting	
	AFShape	Name	shpLine1
	BackColor	&H00AAAAAA&	
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	
	Height	2	
	Left	0	
	Top	45	
	Width	160	
	AFGrid	Name	grdProg
	FontName	AFPalm (12 Regular)	
	FontSize	12	
	FontStyle	0	
	GridLines	1 - Horizontal	
	Height	70	
	Left	0	
	SelectionType	2 - Row	
	Top	70	
	Width	160	
	AFLabel	Name	lblCatgName
	BackColor	&H00FFFFFF&	
	Caption	category name	
	FontName	AFPalm (12 Regular)	
	FontSize	12	
	FontStyle	0	
	Height	12	
	Left	40	
	Top	31	
	Width	120	

Table 20: Lesson 4 - Ingot Properties

The Category Form should now look like this:

Figure 21: frmTVTCatg with Shape, Grid, and Label

Declaring the Program Database

MobileVB™ uses a Long type value to represent a Palm database. Since the program database is accessed throughout the TVToday program, we need to declare a public variable to represent it. To do this, add the following code to the Declarations section of mdlTVTMain:

```
Public glngProg As Long
```

Keeping Track of Program Database Fields

To retrieve information from the Program database, the application code targets specific database records and specific fields within the records. Each field is identified by an integer value.

The TVToday application uses global constants to keep track of each database field. Paste the following code into the Declarations section of mdlTVTMain:

```
'program database fields
Global Const LNG_PROG_PROGRAM As Long = 0
Global Const LNG_PROG_CHANNEL As Long = 1
Global Const LNG_PROG_STARTTIME As Long = 2
Global Const LNG_PROG_ENDTIME As Long = 3
Global Const LNG_PROG_CATGID As Long = 4
Global Const LNG_PROG_DESC As Long = 5
Global Const LNG_PROG_PREVIEW As Long = 6
```

Loading the Programs

To load each corresponding television program into the Category form's grid, a new Sub procedure called LoadPrograms is added to frmTVTCatg. The Sub procedure filters the Program database by category, sorts the Program database by channel, clears and formats the grid, and loads each television program from the database record by record.

Sub LoadPrograms accepts a single argument: the bookmark of the chosen category in the category database. It uses the bookmark to retrieve the chosen category's name and ID (an integer unique to the category). The category name is placed in the user interface. The category ID is used to identify which records in the Program database correspond to the chosen category.

Paste the following code into the code window for frmTVTCatg:

```
Public Sub LoadPrograms(ByRef lngCatgBookmark _
 As Long)
  ’loads programs into grid that are in the
  ’selected category

  Dim intRows As Integer
  Dim lngCatgID As Long
  Dim lngCurrCatgID As Long
  Dim strCatgName As String
  Dim strChannel As String
  Dim strProgram As String

  ’get the selected record’s category name and ID
  PDBFindRecordbyID glngCatg, lngCatgBookmark
  PDBGetField glngCatg, LNG_CATG_CATGID, lngCatgID
  PDBGetField glngCatg, LNG_CATG_CATGNAME, _
   strCatgName
  lblCatgName.Caption = strCatgName

  ’Get the total number of rows in the grid
  intRows = grdProg.Rows

  ’hide the grid so it will draw faster
  grdProg.Visible = False
  ’remove any existing rows from the grid
  While Not (grdProg.Rows = 0)
    grdProg.RemoveItem (0)
    grdProg.Refresh
    intRows = grdProg.Rows
  Wend

  ’format the grid’s column widths
  grdProg.ColWidth(0) = 40
  grdProg.ColWidth(1) = 115

  ’target the left column, so that as cell tag
  ’properties are set below, they are set for the
  ’left cell in each row
  grdProg.Col = 0

  ’start with the first record
  PDBMoveFirst glngProg

  ’while not on the last record of the database
  While Not PDBEOF(glngProg)

    ’get the current program’s category id
    PDBGetField glngProg, LNG_PROG_CATGID, _
     lngCurrCatgID

    ’if the current program’s category id matches
    ’the category being loaded put it in
```

```
      'the program grid
      If lngCurrCatgID = lngCatgID Then
        'get the record's channel and program name
        PDBGetField glngProg, LNG_PROG_CHANNEL, _
         strChannel
        PDBGetField glngProg, LNG_PROG_PROGRAM, _
         strProgram

        'add as a new program in the grid
        grdProg.AddItem strChannel & Chr(9) _
         & strProgram, -1

        'select the row just added to the grid
        grdProg.Row = (grdProg.Rows - 1)
        'set the cell's tag as the current
        'record's bookmark
        grdProg.ItemData(grdProg.Row, grdProg.Col) _
         = PDBRecordUniqueID(glngProg)

        'set the cell's current font
        grdProg.FontStyle = 1
      End If

      'move to the next record in the database
      PDBMoveNext glngProg
    Wend

    'If no rows are in the grid, add a row indicating
    'there are no listings for this time slot and
    'disable the show me button.
    'Otherwise enable the show me button
    If grdProg.Rows = 0 Then
      grdProg.AddItem "No" & Chr(9) & "Listings"
      btnShowMe.Enabled = False
      btnShowMe.ForeColor = afButtonLightGray
    Else
      btnShowMe.Enabled = True
      btnShowMe.ForeColor = afButtonBlack
    End If

    'select the first grid item
    grdProg.Row = 0
    'show the grid
    grdProg.Visible = True
    grdProg.Refresh

  End Sub
```

Note that as each program is added as a new row to the grid, the new row's left column is tagged with the program's record unique ID. This tag will be used later to retrieve more information on the program.

Calling Sub LoadPrograms

The corresponding television programs should be loaded into the grid before the user sees the Category form. Therefore, the Sub procedure LoadPrograms should be called from the Show Me button's Click event on the Main form.

In addition, the database must be opened and a schema must be defined for the database prior to calling LoadPrograms. If the database is not opened the LoadProgram Sub procedure will generate errors. The schema provides a means to define and label the fields in the database.

Edit the Show Me button's Click event procedure in frmTVTMain so it matches the following (the bold code indicates the newly added lines):

```
Private Sub btnShowMe_Click()
  'load the programs into the category form's grid
  'that correspond to the selected category
  Call frmTVTCatg.LoadPrograms(CLng( _
   lstCatg.ItemData(lstCatg.ListIndex)))
  'hide this form and show the category form
  frmTVTMain.Hide
  frmTVTCatg.Show
End Sub
```

Opening the Program Database

Prior to calling LoadCategories, the Program database must be opened. Otherwise, the database will remain closed and the LoadCategories Sub procedure will generate errors. Additionally, a successful open of the database should be verified. This is done by adding the program database to the existing check of the category database that already exists in Sub Main.

Edit the Sub Main in the standard module mdlTVTMain so it matches the following (the bold code indicates the newly added or edited lines):

```
Sub Main()
  Dim strCatgPath As String
  Dim strProgPath As String

  'Use conditional compilation to determine if
  'program is running on Palm or Windows
  #If APPFORGE Then
    'Palm Path
    strCatgPath = "Category"
    strProgPath = "Program"

  #Else
    'Windows Path
    strCatgPath = App.Path & "\Category"
    strProgPath = App.Path & "\Program"

  #End If

  'open the databases
  glngCatg = PDBOpen(Byfilename,strCatgPath, 0, _
```

```
     0, 0, 0, 0)
    glngProg = PDBOpen(Byfilename, strProgPath, 0, _
     0, 0, 0, 0)

    'Check to see if the databases were
    'successfully opened
    'If not bring up the error form
    'If so continue by setting the schema
    'and loading the categories
    If glngCatg = 0 Or glngProg = 0 Then
      'Database was not successfully opened

      'Run the Load Error subroutine
      'and show frmTVTError
      frmTVTError.LoadError
      frmTVTError.Show

    Else
      'load the categories from the database into
      'the Main form's listbox
      frmTVTMain.LoadCategories

      'show form
      frmTVTMain.Show

    End If

  End Sub
```

There are two important points to note about the path argument of the PDBOpen method:

1. The path argument for the PDBOpen method is different when running in Palm OS and running in Windows. Conditional compilation is used to ensure the proper path is assigned for each.
2. When running on a Palm device, the database name is **case-sensitive.** Make sure that the name is typed in the correct case.

Editing the LoadError Subroutine

TVToday now accesses two databases. Therefore, the error message shown when a database is not successfully opened must identify which database has a problem. The error message is assembled in the LoadError subroutine of frmTVTError. Edit it so that it matches the following (the bold code indicates the newly added or edited lines):

```
  Public Sub LoadError()
    Dim strDbName As String

    'Identify the missing database
    If glngCatg = 0 Then
      strDbName = "Category.PDB"

      If glngProg = 0 Then
```

```
        strDbName = strDbName & " and Program.PDB"
      End If

    ElseIf glngProg = 0 Then
      strDbName = "Program.PDB"

    End If

    'Set the text for the Error Message text box
     txtErrorMsg.Text = strDbName & _
     " not found. Please HotSync " & strDbName & _
     " and run TVToday again."

End Sub
```

Running the Project

Save the changes made to the project by selecting Save Project from the File menu.

Run the application by choosing **Start** from the Run menu, or by clicking the Start button on the toolbar. The main form will appear with a list of categories within the AFListBox. Click on the Show Me button to navigate to the Category form. A list of matching television programs should be displayed on the grid.

Lesson 5: Displaying Programs by the Hour

The TVToday application should only display television programs in the Category form that air within a one-hour timeframe. Currently, the Sub procedure LoadPrograms loads all the television programs into the grid, regardless of the program's start or end time. To create this functionality, additional code is added to the Sub procedure LoadPrograms, so that the display will look like the following figure.

Figure 22: Category Screen of the TVToday Application

Adding Time-Based Code

To make the grid only display a one-hour timeframe of programs, we need to add code that filters the data to the Sub procedure LoadPrograms. If a program meets any of the following criteria, the program's data is inserted into the grid:

- The start time for the program is on or within the current hour.
- The end time for the program is within or at the end the current hour.
- If the start time for the program is before the current hour and the end time for the program is after the current hour.

The following figure illustrates the "filtering" logic.

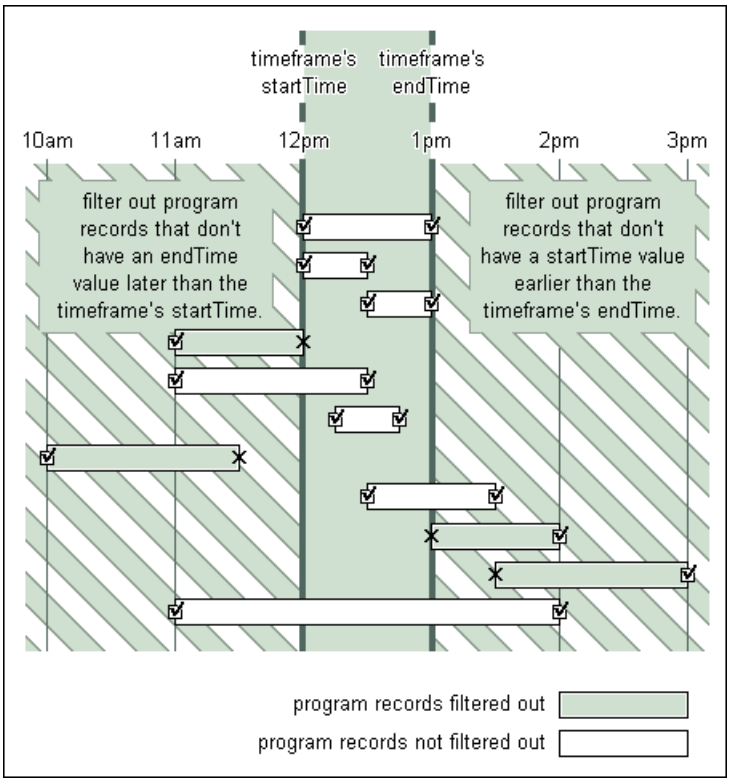

Figure 23: Illustration Time-Based Filter Logic

The beginning of the one-hour timeframe will be determined by a second procedure argument that should be added to the Sub LoadPrograms.

Edit the LoadPrograms Sub procedure in frmTVTCatg so it matches the following (the bold code indicates the newly added lines):

```
Public Sub LoadPrograms(ByVal datNewTime As Date, _
 ByRef lngCatgBookmark As Long)

  'loads programs into grid that are in the selected
  'category, and are viewable between the selected
  'new time and an hour past the new time

  Dim datMaxTime As Date
  Dim datMinTime As Date
  Dim datStartTime As Date
  Dim datEndTime As Date
  Dim intRows As Integer
  Dim lngCatgID As Long
  Dim lngCurrCatgID As Long
  Dim strCatgName As String
  Dim strChannel As String
  Dim strProgram As String
  Dim blnFillin As Boolean
```

```
’get the selected record’s category name and ID
PDBFindRecordbyID glngCatg, lngCatgBookmark
PDBGetField glngCatg, LNG_CATG_CATGID, lngCatgID
PDBGetField glngCatg, LNG_CATG_CATGNAME, _
 strCatgName
lblCatgName.Caption = strCatgName

’calculate the timeframe’s start time
datMinTime = datNewTime
’calculate the timeframe’s end time
datMaxTime = DateAdd("h", 1, datNewTime)

’Get the total number of rows in the grid
intRows = grdProg.Rows

’hide the grid so it will draw faster
grdProg.Visible = False
’remove any existing rows from the grid
While Not (grdProg.Rows = 0)
  grdProg.RemoveItem (0)
  grdProg.Refresh
  intRows = grdProg.Rows
Wend

’format the grid’s column widths
grdProg.ColWidth(0) = 40
grdProg.ColWidth(1) = 115

’target the left column, so that as cell tag
’properties are set below, they are set for the
’left cell in each row
grdProg.Col = 0

’start with the first record
PDBMoveFirst glngProg

’while not on the last record of the database
While Not PDBEOF(glngProg)

  ’get the current program’s category id
  PDBGetField glngProg, LNG_PROG_CATGID, _
   lngCurrCatgID

  ’if the current program’s category id matches
  ’the category being loaded
  If lngCurrCatgID = lngCatgID Then

    ’Get the start and end times for the programs
    ’and convert them to time only format
    PDBGetField glngProg, LNG_PROG_STARTTIME, _
     datStartTime
```

```
  datStartTime = TimeSerial(Hour(datStartTime) _
   , Minute(datStartTime), 0)
  PDBGetField glngProg, LNG_PROG_ENDTIME, _
   datEndTime
  datEndTime = TimeSerial(Hour(datEndTime), _
   Minute(datEndTime), 0)

  'reset blnFillin to false
  blnFillin = False

  'If the start time is on or within the current
  'hour, set blnFillin to true
  If (datStartTime >= datMinTime) And _
   (datStartTime   < datMaxTime) Then _
    blnFillin = True
  'If the end time is within the or at the end
  'current hour set blnFillin to true
  If (datEndTime > datMinTime) And (datEndTime _
   <= datMaxTime) Then blnFillin = True
  'If the start time is before the current hour
  'and the end time is after the current hour
  'set blnFillin to true
  If (datStartTime < datMinTime And datEndTime _
   > datMaxTime) Then blnFillin = True

  'If any of the above criteria were met place
  'program data in the grid
  If blnFillin Then

    'get the record's channel and program name
    PDBGetField glngProg, LNG_PROG_CHANNEL, _
     strChannel
    PDBGetField glngProg, LNG_PROG_PROGRAM, _
     strProgram

    'add as a new program in the grid
    grdProg.AddItem strChannel & Chr(9) _
     & strProgram, -1

    'select the row just added to the grid
    grdProg.Row = (grdProg.Rows - 1)
    'set the cell's tag as the current
    'record's bookmark
    grdProg.ItemData(grdProg.Row, grdProg.Col) = _
    PDBRecordUniqueID(glngProg)

    'set the cell's current font
    grdProg.FontStyle = 1

  End If

End If
```

```
    'move to the next record in the database
    PDBMoveNext glngProg

  Wend

  'If no rows are in the grid, add a row indicating
  'there are no listings for this time slot and
  'disable the show me button.
  'Otherwise enable the show me button
  If grdProg.Rows = 0 Then
    grdProg.AddItem "No" & Chr(9) & "Listings"
    btnShowMe.Enabled = False
    btnShowMe.ForeColor = afButtonLightGray
  Else
    btnShowMe.Enabled = True
    btnShowMe.ForeColor = afButtonBlack
  End If

  'select the first grid item
  grdProg.Row = 0
  'show the grid
  grdProg.Visible = True
End Sub
```

Notice that the start and end times were truncated to only contain time. When the date values are extracted from the database, they have a date associated with them. This would cause the time comparisons to function improperly.

Determining the Top of the Hour

When users choose to view the television programs corresponding to a specific category, the Category form should display the programs that can be viewed during the current hour. For example, if the time is 2:43pm, the Category form should display the television programs that can be viewed between 2:00pm and 3:00pm.

To support this functionality, we will add a new function called TopOfTheHour to the standard module mdlTVT-Main. This function receives a date argument containing a time value, truncates the time value down to the closest even hour, and returns the truncated time value as a date. Paste the following code into the standard module mdlTVTMain:

```
Public Function TopOfTheHour(ByVal datNewTime _
 As Date) As Date
  'returns a date containing the date argument
  'rounded down to the previous even
  'hour (e.g., 4:53am => 4:00am)

  'remove minutes past the hour
  datNewTime = DateAdd("n", (-1 * _
   (Minute(datNewTime))), datNewTime)
  'remove seconds past the hour
  datNewTime = DateAdd("s", (-1 * _
   (Second(datNewTime))), datNewTime)
```

```
  'return the rounded down date
  TopOfTheHour = datNewTime
End Function
```

Calling the Revised LoadPrograms

Now that the Sub procedure LoadPrograms requires a date argument, the code calling LoadPrograms from frmTVTMain must be updated. The updated code in the Show Me button's Click event will determine the top of the current hour and pass that value as an argument to the Sub procedure LoadPrograms. Edit the Show Me button's Click event procedure on the Main form so it matches the following (the bolded code indicates newly added or modified lines):

```
Private Sub btnShowMe_Click()
  'load the programs into the category
  'form's grid that
  'correspond to the selected category and are
  'on during the current hour

  Dim datThisHour As Date

  'get the top of the current hour
  datThisHour = mdlTVTMain.TopOfTheHour(Time)

  'load programs
  Call frmTVTCatg.LoadPrograms(datThisHour, _
  CLng(lstCatg.ItemData(lstCatg.ListIndex)))

  'hide this form and show the category form
  frmTVTMain.Hide
  frmTVTCatg.Show
End Sub
```

Running the Project

Save the changes made to the project by selecting Save Project from the File menu.

Run the application by choosing **Start** from the Run menu, or by clicking the Start button on the toolbar. Click on the Show Me button to navigate to the Category form. A much smaller list of matching television programs is displayed in the grid, due to the timeframe filters. Some categories may not have any matching programs for the current timeframe.

Lesson 6: Providing Control Over the Program Timeframe

The Category form should show the value of the current one-hour timeframe being displayed and give the users the capability to change that timeframe. To accomplish this we will add several Ingots to the Category form, add code to Sub LoadPrograms to display the current timeframe in the user interface, and create code that gives the users control over the timeframe. Following this lesson, the Category Screen should look like the following figure.

Figure 24: Category Screen of the TVToday Application

To support user control of the currently displayed one-hour timeframe, the following AppForge Ingots will be added during Lesson 6:

Button	Ingot	Description
	AFShape	Used to create rectangles or circles on your forms.
	AFLabel	Functions as a text field that is not directly editable by the user.
	AFGraphicButton	Works like the standard button, but instead of a caption, it can specify different graphics for the up and down click positions.

Table 21: Lesson 6 - Ingots to Add

Adding Ingots to Control the Timeframe

Use the toolbox to draw an AFShape, AFLabel, and two AFGraphicButton Ingots on frmTVTCatg. In the Properties window, set properties for the each Ingot according to the following table. Use the default setting for all other properties.

Ingot	Property	Setting	
	AFShape	Name	shpRect5
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	

	FillStyle	0 - Solid	
	Height	22	
	Left	95	
	Top	47	
	Width	124	
			
	AFLabel	Name	lblHour
	Alignment	2 - Center	
	BackColor	&H00FFFFFF&	
	Caption	12:00am	
	FontName	AFPalm (12 Regular)	
	FontSize	12	
	FontStyle	0	
	Height	10	
	Left	27	
	Top	53	
	Width	45	
			
	AFGraphicButton1	Name	grbTimeBak
	DisabledPicture	TVT_ARBD.RGX	
	DownPicture	TVT_ARBH.RGX	
	FocusPicture	TVT_ARBK.RGX	
	Height	22	
	Left	0	
	NoFocusPicture	TVT_ARBK.RGX	
	Top	47	
	Width	22	
			
	AFGraphicButton2	Name	grbTimeFwd
	DisabledPicture	TVT_ARFD.RGX	
	DownPicture	TVT_ARFH.RGX	
	FocusPicture	TVT_ARFW.RGX	
	Height	22	
	Left	73	
	NoFocusPicture	TVT_ARFW.RGX	
	Top	47	
	Width	22	

Table 22: Lesson 6 - New Ingot Properties

If all of the properties have been set properly, frmTVTCatg should look like the following figure.

Figure 25: frmTVTCatg with Shape, Label, and GraphicButtons

Displaying and Retaining the Current Timeframe Value

TVToday needs to display the current timeframe value for user feedback, and it also must retain the current timeframe value in global memory for tracking. To do this, two new module-wide variables are used to retain the time value and the category bookmark. They are declared in the form module of frmTVTCatg.

Paste the following code into the Declarations section of frmTVTCatg:

```
'stores current hour shown in UI
Dim datCurTime As Date
'stores the database bookmark of the currently
'displayed category
Dim lngCurCatgBookmark As Long
```

Next, append code to the Sub procedure LoadPrograms in frmTVTCatg to allow it to retain and display the current timeframe value and category bookmark. Edit Sub LoadPrograms so the last lines of code prior to End Sub appear as follows (the bold code indicates newly added or modified lines):

```
    'select the first grid item
    grdProg.Row = 0
    'show the grid
    grdProg.Visible = True
    grdProg.Refresh

    'show the start time in the UI
    lblHour.Caption = TimeToStr(datNewTime)
    'store the start time
    datCurTime = datNewTime
    'store the database bookmark of the currently
    'displayed category
    lngCurCatgBookmark = lngCatgBookmark

End Sub
```

Finally, we need to add a function to convert a time value (stored in a date variable) to a formatted string variable (for display in the user interface). Paste the following code into the standard module mdlTVTMain:

```
Public Function TimeToStr(ByVal datNewTime As Date) _
As String
  'returns a string representing the date argument
  '(e.g., #12:15 PM# => "12:15pm")

  Dim intHour As Integer
  Dim intMinute As Integer
  Dim strMinute As String
  Dim str_M As String

  'get the hour and minute values
  intHour = Hour(datNewTime)
  intMinute = Minute(datNewTime)

  'determine if it's AM or PM
  If (intHour < 12) Then
    str_M = "am"
  Else
    str_M = "pm"
  End If

  'adjust the hour from 24 hour time
  If (intHour > 12) Then
    intHour = (intHour - 12)
  ElseIf (intHour = 0) Then
    intHour = 12
  End If

  'zero pad one digit minute values
  If (intMinute < 10) Then
    strMinute = "0" & Trim(Str(intMinute))
  Else
    strMinute = Trim(Str(intMinute))
  End If

  'return the results
  TimeToStr = Trim(Str(intHour)) & ":" & _
   strMinute & str_M
End Function
```

Notice that the Sub procedure TimeToStr was called from the Sub procedure LoadPrograms to build a string value of the current time for display in the user interface.

Allowing Users to Control the Timeframe

To provide users with direct control of the current timeframe, we need to associate code with the Click events of both AFGraphicButton Ingots created earlier in this lesson. In each event procedure, the Sub procedure LoadPrograms is called to reload the grid with a new timeframe of programs.

Double-click on the grbTimeBak to view its Click event procedure in the code window. Paste the following code into frmTVTCatg's form module:

```
Private Sub grbTimeBak_Click()
  'show the previous hour's programs in the UI
  Call LoadPrograms(DateAdd("h", -1, datCurTime), _
   lngCurCatgBookmark)
End Sub
```

Notice how the time argument supplied to the Sub procedure LoadPrograms is the stored current timeframe value decreased by an hour. The bookmark argument supplied to the Sub procedure LoadPrograms is the value stored by the Sub procedure LoadPrograms the last time it was called.

Next, double-click on grbTimeFwd to view its Click event procedure in the Code window. Paste the following code into frmTVTCatg's form module:

```
Private Sub grbTimeFwd_Click()
  'show the next hour's programs in the UI
  Call LoadPrograms(DateAdd("h", 1, datCurTime), _
  lngCurCatgBookmark)
End Sub
```

Finally, add code to disable grbTimeFwd and grbTimeBak if the last hour or first hour of the day has been reached. Edit Sub LoadPrograms in frmTVTCatg so the last lines of code prior to End Sub appear as follows:

```
  'show the start time in the UI
  txtHour.Text = TimeToStr(datNewTime)
  'store the start time
  datCurTime = datNewTime
  'store the database bookmark of the currently
  'displayed category
  lngCurCatgBookmark = lngCatgBookmark

  'disable the forward graphicButton depending on
  'the value of the start time
  If (Hour(datCurTime) = 23) Then
    grbTimeFwd.Enabled = False
  Else
    grbTimeFwd.Enabled = True
  End If

  'disable the backward graphicButton depending on
  'the value of the start time
  If (Hour(datCurTime) = 0) Then
    grbTimeBak.Enabled = False
  Else
    grbTimeBak.Enabled = True
  End If

End Sub
```

Running the Project

Save the changes made to the project and run the application. Select a category in the AFListBox. Click on the Show Me button to navigate to the Category form. The form displays a list of matching television programs in

the grid, as well as the start of the current one-hour timeframe. Click the AFGraphicButtons to the left and right of the current timeframe value to change the timeframe.

Lesson 7: Creating a Program Form

In this lesson, we will add a new form to display detailed information on a selected television program. Once users have selected a program on the Category form, we want them to be able to click the Show Me button to view detailed program information on a third form.

Figure 26: frmTVTProg with AFShape, AFTextBox, and AFGraphicButtons

To create the user interface for the Program form, AppForge Ingots are added to the form. Lesson 7 of the tutorial uses the following Ingots:

Button	Ingot	Description
	AFShape	Creates rectangles or circles on your forms.
	AFButton	Text button that can receive user events to begin, interrupt, or end a process.
	AFGraphic	Displays a graphic in the application. Valid graphic formats are .bmp, .rgx, .jpg, or .png*.
	AFLabel	Functions as a text field that is not directly editable by the user.
	AFGraphicButton	Works like the standard button, but instead of a caption, it can specify different graphics for the up and down click positions.

Table 23: Lesson 7 - Ingots to Add

** AppForge MobileVB 3.1 or later is required to utilize a .png file. See the Crossfire Client and PNG Files Section of the AppForge Crossfire Client And Additional Files documentation for details.*

Adding a New Form

Before laying out the user interface for the Program form, a new form must be added to the project.

1. Select **Add Form** from the Project menu. This will bring up the Add Form dialog box.
2. Select Form from the Add Form dialog box.
3. Click the **Open** button.

In order to identify this form later, it should be named. In the Properties window, set the form's name property to **frmTVTProg** and clear the Caption property.

Copying Ingots to the Program Form

Nine of the Ingots we will use on the Program form are nearly identical to Ingots on the Category form. To save time, copy the matching Ingots from frmTVTCatg to frmTVTProg. The nine matching Ingots are named grbTVBack, lblTVToday, lblCatgName, shpLine1, shpRect2, shpRect3, shpRect4, btnShowMe, and gphArrowRight.

To copy Ingots from one form to another:

1. Open the Category form, frmTVTCatg. You can double-click on the form's name in the Project Explorer window to show it.
2. Select the first component to copy by clicking on it in the form.
3. Select each additional component to copy by holding down the Shift key and clicking on each component you want.
4. Copy the components by selecting Copy from the Edit menu, or by clicking the Copy button on the toolbar.
5. Open the Ingots' destination form, frmTVTProg.
6. Paste the components by selecting Paste from the Edit menu, or by clicking the Paste button on the toolbar.

Once the AppForge Ingots are pasted onto the Program form, update two Ingots' properties:

Ingot	Property	Setting
shpLine1	Left	21
	Top	48
	Width	139
btnShowMe	Name	btnPreview
	Caption	preview

Table 24: Copied Ingots Properties

The Program Form should now look like this:

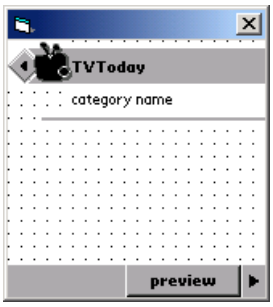

Figure 27: Program Form with Copied Ingots

Adding Ingots to the Program Form

Use the toolbox to draw an AFShape and AFGraphicButton on frmTVTProg. In the Properties window, set properties for the Ingots according to the following table. Use the default setting for all other properties.

Ingot	Property	Setting	
	AFShape	Name	shpLine2
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	
	FillStyle	0 - Solid	
	Height	2	
	Left	0	
	Top	72	
	Width	160	
	AFGraphicButton	Name	grbCatgBack
	DownPicture	TVT_CGBH.RGX	
	FocusPicture	TVT_CGBK.RGX	
	Height	23	
	Left	0	
	NoFocusPicture	TVT_CGBK.RGX	
	Top	27	
	Width	21	

Table 25: Lesson 7 - New Ingot Properties

The Program Form should now look like the following figure.

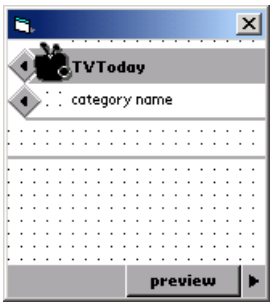

Figure 28: Program Form with Shape and GraphicButton

Creating Navigation between Category and Program Forms

When users click the Show Me button on the Category form, the Program form should replace the Category form. Double-click the Show Me button in frmTVTCatg to view the button's Click event in the Code window. Paste the following code into frmTVTCatg's form module:

```
Private Sub btnShowMe_Click()
  'hide this form and show the program form
  frmTVTCatg.Hide
  frmTVTProg.Show
End Sub
```

To complete the navigation between the Category and Program forms, code must be added to frmTVTProg. When users click on the lower AFGraphicButton on the Program form, the Category form should replace the Program form. Double-click the lower AFGraphicButton on the Program form to view grbCatgBack's Click event in the Code window. Paste the following code into frmTVTProg's form module:

```
Private Sub grbCatgBack_Click()
  'hide this form and show the category form
  frmTVTProg.Hide
  frmTVTCatg.Show
End Sub
```

Creating Navigation from Program Form to Main Form

When the top AFGraphicButton on the Program Form is clicked, the Main form should replace the Program form. Double-click the top AFGraphicButton on the Program form to view grbTVBack's Click event in the code window. Paste the following code into frmTVTProg's form module:

```
Private Sub grbTVBack_Click()
  'hide this form and show the main form
  frmTVTProg.Hide
  frmTVTMain.Show
End Sub
```

Saving the Project

Save the changes made to the project by selecting Save Project from the File menu. Save the following file with the corresponding file name:

File	File Name
Form	frmTVTProg.frm

Table 26: Lesson 7 - New Form Name

Running the Project

Run the application. Select a category in the ListBox and click on the Show Me button to navigate to the Category form. Select a program in the grid and click on the Show Me button again to see the program form. You can then click on either graphic button to navigate to one of the previous two forms.

Lesson 8: Displaying Program Information

When users choose to view a program on the Program form, we want the form to display the title, the channel, the starting and ending times, and a description of the program, if available.

Figure 29: Program Screen of the TVToday Application

In this lesson, program information is loaded from the Program database into the user interface. Begin by adding AppForge Ingots to the Program form that will display the program information. Lesson 8 of the tutorial uses the following Ingots:

Button	Ingot	Description
	AFLabel	Functions as a text field that is not directly editable by the user.
	AFTextBox	Displays text on the form, or provides a text input box for the user.

Table 27: Lesson 8 - Ingots to Add

Adding Display Ingots to the Program Form

Use the toolbox to draw one AFTextBox and one AFLabel on frmTVTProg. In the Properties window, set properties for the each Ingot according to the following table. Use the default setting for all other properties.

Ingot	Property	Setting	
	AFLabel	Name	lblProgName
	BackColor	&H00FFFFFF&	
	Caption	Program Name	
	FontName	AFPalm (15 Regular)	
	FontSize	15	
	FontStyle	0	
	Height	15	

	Left	7	
	Top	55	
	Width	153	
			
	AFTextBox	Name	txtDesc
	BackColor	&H00FFFFFF&	
	BorderStyle	1 - Single	
	FontName	AFPalm (12 Regular)	
	FontSize	12	
	FontStyle	0	
	Height	66	
	Left	0	
	MultiLine	True	
	ScrollBars	2 - Vertical	
	Text	This is the description text for a television program.	
	Top	74	
	UnderlineStyle	0 - None	
	Width	160	

Table 28: Lesson 8 - Ingot Properties

The Program Form should now look the following figure.

Figure 30: Program Form with Label and TextBox

Formatting the Time Range

Two points of information supplied to users on the Program form are the start time and the end time of the selected program. The data for these reside in the Program database in date-type data fields. To present this information in the user interface, the time values have to be converted to strings and formatted.

This calls for the addition of a new function to the standard module mdlTVTMain. This new function, named TimeRangeToStr, will accept two time value arguments as dates and return a formatted string containing the time

values.

Paste the following code into the standard module mdlTVTMain:

```
Public Function TimeRangeToStr(ByVal datStartTime _
 As Date, ByVal datEndTime As Date) As String
  'returns a string defining a time range based
  'on the two date arguments (e.g., #12:30 PM# and
  '#1:30 PM# => "12:30 - 1:30pm")

  Dim strStartTime As String
  Dim strEndTime As String

  'convert the date arguments to strings
  strStartTime = TimeToStr(datStartTime)
  strEndTime = TimeToStr(datEndTime)

  'if the times have the same AM/PM
  If (Right(strStartTime, 2) = Right(strEndTime, 2)) _
   Then
    'remove the AM/PM from the start time
    strStartTime = Left(strStartTime, _
     (Len(strStartTime) - 2))
  End If

  'build and return the date range
  TimeRangeToStr = strStartTime & " - " & strEndTime
End Function
```

Note that this Sub procedure calls the Sub procedure TimeToStr to convert the time values from dates to strings.

Loading the Program Information

To load program information into the Program form, a new Sub procedure called LoadProgramInfo needs to be added to frmTVTProg. This Sub procedure displays the current category name in the user interface, finds the selected program in the Program database, and retrieves and displays program information from the record.

Paste the following code into the code window for frmTVTProg:

```
Public Sub LoadProgramInfo(ByVal lngProgBookmark _
 As Long, ByVal strCatgName As String)
  'loads program information into the UI that
  'corresponds to the selected program bookmark

  Dim strChannel As String
  Dim strProgram As String
  Dim strDescription As String
  Dim datStartTime As Date
  Dim datEndTime As Date
  Dim strProgTime As String

  'put the category name in the UI
```

```
  lblCatgName.Caption = strCatgName

  'target the selected program
  PDBFindRecordbyID glngProg, lngProgBookmark

  'put the program name in the UI
  PDBGetField glngProg, LNG_PROG_PROGRAM, strProgram
  lblProgName.Caption = strProgram

  'get the channel
  PDBGetField glngProg, LNG_PROG_CHANNEL, strChannel

  'get the start time
  PDBGetField glngProg, LNG_PROG_STARTTIME, _
   datStartTime
  datStartTime = TimeSerial(Hour(datStartTime), _
    Minute(datStartTime), Second(datStartTime))

  'get the end time
  PDBGetField glngProg, LNG_PROG_ENDTIME, datEndTime
  datEndTime = TimeSerial(Hour(datEndTime), _
    Minute(datEndTime), Second(datEndTime))

  'get the description
  PDBGetField glngProg, LNG_PROG_DESC, _
   strDescription

  'build the date range string
  strProgTime = TimeRangeToStr(datStartTime, _
   datEndTime)

  'put the description in the UI
  txtDesc.Text = strChannel & ": " & strProgTime & _
   Chr(13) & strDescription
End Sub
```

Notice how this Sub procedure calls the Sub procedure TimeRangeToStr to build the date range string.

Calling Sub LoadProgramInfo

The program information should be loaded into the Program form's user interface before the Program form is shown. Therefore, the Sub procedure LoadProgramInfo should be called in the Click event of the Show Me button on frmTVTCatg.

Double-click the Show Me button on the Category form to view its Click event procedure. Edit the procedure so that it matches the following (the bold code indicates newly added or modified lines):

```
Private Sub btnShowMe_Click()
  'load the program info into the program form's UI
  'that correspond to the selected program

  'target the gridrow's left cell to retrieve
```

```
    'its tag (i.e., the corresponding bookmark)
    grdProg.Col = 0

    'load the programs
    Call frmTVTProg.LoadProgramInfo(grdProg.ItemData _
     (grdProg.Row, grdProg.Col), lblCatgName.Caption)

    'hide this form and show the program form
    frmTVTCatg.Hide
    frmTVTProg.Show
End Sub
```

Notice how this procedure references a program's database record by retrieving the record's unique ID from the grid. In Lesson 4, the unique ID values were added as tags to the left cell of every new grid row.

Running the Project

Save the changes made to the project and run the application. Select a category in the ListBox. Click on the Show Me button to navigate to the Category form, and select a program in the grid. Click on the Show Me button to display the program's category, title, channel, start time, end time, and description.

Lesson 9: Creating the Preview Form

In this lesson, a new form is added to display filmstrip previews of a selected television program. When users navigate to the Program form, we want the Preview button to be enabled or disabled depending on the availability of a filmstrip preview for the selected TV program. If a filmstrip preview is available, users may click on the Preview button to watch it.

Figure 31: Preview Screen of the TVToday Application

To create the user interface for the Preview form, Lesson 9 of the tutorial uses the following AppForge Ingots:

Button	Ingot	Description
	AFShape	Used to create rectangles or circles on your forms.
	AFLabel	Functions as a text field that is not directly editable by the user.
	AFGraphicButton	Works like the standard button, but instead of a caption, it can specify different graphics for the up and down click positions.

Table 29: Lesson 9 - Ingots to Add

Adding a Preview Form

Before laying out the user interface for the Preview form, a new form must be added to the project.

1. Select **Add Form** from the Project menu. This will bring up the Add Form dialog box.
2. Select Form from the Add Form dialog box.
3. Click the **Open** button.

In the Properties window, set the form's name property to **frmTVTPrev** and clear the Caption property.

Copying Ingots to the Preview Form

Nine of the Ingots we will use on the Preview form are nearly identical to Ingots on the Program form. To save time, copy the matching Ingots from the Program form to the Preview form. The nine matching Ingots are named: grbTVBack, lblTVToday, grbCatgBack, lblCatgName, shpLine1, shpLine2, shpRect3, shpRect4, and lblProgName.

To copy Ingots from one form to another:

1. Open the Program form, frmTVTProg. You can double-click on the form's name in the Project Explorer window to show it.
2. Select the first component to copy by clicking on it in the form.
3. Select each additional component to copy by holding down the Shift key and clicking on each component you want.
4. Copy the components by selecting Copy from the Edit menu, or by clicking the Copy button on the toolbar.
5. Open the Ingots' destination form, frmTVTPrev.
6. Paste the components by selecting Paste from the Edit menu, or by clicking the Paste button on the toolbar.

Once the AppForge Ingots are pasted onto the Preview form, change two Ingots' properties to the following:

Ingot	Property	Setting
lblProgName	Left	40
	Width	120
shpLine2	Left	21
	Width	139

Table 30: Lesson 9 - Copied Ingots Property Changes

The following figure shows the Preview Form as it should now appear.

Figure 32: Preview Form with Copied Ingots

Adding Ingots to the Preview Form

Use the toolbox to draw an AFGraphicButton Ingot and four AFShape Ingots on frmTVTCatg. In the Properties window, set properties for the Ingots according to the following table. Keep all other settings at their defaults.

Ingot	Property	Setting	
			
	AFGraphicButton	Name	grbProgBack
	DownPicture	TVT_PGBH.RGX	
	FocusPicture	TVT_PGBK.RGX	
	Height	23	
	Left	0	
	NoFocusPicture	TVT_PGBK.RGX	
	Top	51	
	Width	21	
			
	AFShape1	Name	shpRect5
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	
	FillStyle	0 - Solid	
	Height	4	
	Left	21	
	Top	156	
	Width	139	
			
	AFShape2	Name	shpRect6
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	
	FillStyle	0 - Solid	
	Height	84	
	Left	0	
	Top	76	
	Width	21	
			
	AFShape3	Name	shpRect7
	BorderStyle	0 - Transparent	
	FillColor	&H00AAAAAA&	
	FillStyle	0 - Solid	
	Height	80	
	Left	141	
	Top	76	
	Width	19	

AFShape4	Name	shpRect8
BorderStyle	0 - Transparent	
FillColor	&H00AAAAAA&	
FillStyle	0 - Solid	
Height	3	
Left	0	
Top	73	
Width	160	

Table 31: Lesson 9 - New Ingot Properties

The Preview Form should now look like this:

Figure 33: frmTVTPrev with AFGraphicButton and Shape Ingots

Enabling and Disabling the Preview Button

Not every TV program listed in the Program database will have a filmstrip preview. Therefore, we need to add code that checks for the availability of a preview and enables or disables the Preview button accordingly. Edit the beginning of the Sub procedure LoadProgramInfo in the frmTVTProg code window so that it includes the following code (the bold code indicates the newly added line):

```
Public Sub LoadProgramInfo(ByVal lngProgBookmark _
 As Long, ByVal strCatgName As String)
  'loads program information into the UI that
  'corresponds to the selected program bookmark

  Dim strChannel As String
  Dim strProgram As String
  Dim strDescription As String
  Dim strPreview As String
  Dim datStartTime As Date
  Dim datEndTime As Date
  Dim strProgTime As String
```

Now edit the end of LoadProgramInfo (newly added lines in bold):

```
  'put the description in the UI
  txtDesc.Text = strChannel & ": " & strProgTime _
   & Chr(13) & strDescription

  'if there's not a preview for the program
  PDBGetField glngProg, LNG_PROG_PREVIEW, _
   strPreview
  If (strPreview = "") Then
    'disable the preview button
    btnPreview.Enabled = False
  Else
    'enable the preview button
    btnPreview.Enabled = True
  End If

End Sub
```

Creating Navigation between Program and Preview Forms

When a filmstrip preview is available for a TV program, we want users to be able to click the Preview button to view the filmstrip clip. The Preview form should replace the Program form when they click the button.

Double-click the Preview button on the Program form to view the button's Click event in the code window. Paste the following code into frmTVTProg's form module:

```
Private Sub btnPreview_Click()
  'hide this form and show the preview form
  frmTVTProg.Hide
  frmTVTPrev.Show
End Sub
```

To complete the navigation between the Program and Preview forms, we need to add code to frmTVTPrev. When users click on the bottom AFGraphicButton on the Preview form, the Program form should replace the Preview form. Paste the following code into frmTVTPrev's form module:

```
Private Sub grbProgBack_Click()
  'hide this form and show the program form
  frmTVTPrev.Hide
  frmTVTProg.Show
End Sub
```

Creating Navigation from Preview Form to Category Form

When users click on the AFGraphicButton next to the category name on the Preview form, the Category form should replace the Preview form. Paste the following code into frmTVTPrev's form module:

```
Private Sub grbCatgBack_Click()
  'hide this form and show the category form
  frmTVTPrev.Hide
  frmTVTCatg.Show
End Sub
```

Creating Navigation from Preview Form to Main Form

When users click on the top AFGraphicButton on the Preview form, the Main form should replace the Preview form. Paste the following code into frmTVTPrev's form module:

```
Private Sub grbTVBack_Click()
  'hide this form and show the main form
  frmTVTPrev.Hide
  frmTVTMain.Show
End Sub
```

Running the Project

Save and run the application. Select the Drama category in the ListBox. Click on the Show Me button to navigate to the Category form, and move to the 2:00pm timeframe. Select "Before the Storm" in the grid. Click on the Show Me button to view the program, and click on the Preview button to move to the Preview form. Click on any of the three AFGraphicButtons to navigate to one of the previous three forms.

Lesson 10: Displaying Program Previews

When users choose to view a program's preview, the Preview form should appear containing the program's filmstrip clip. The program database provides a field that indicates the existence of a preview. The preview should play continuously until the user navigates away from the Preview form.

Figure 34: Preview Screen of the TVToday Application

In this lesson, filmstrip previews are loaded into the user interface using the corresponding filenames in the Program database. In Lesson 10, an AppForge Filmstrip Ingot is added to the Preview form:

Button	Ingot	Description
	AFFilmstrip	Allows the application designer to animate a series of graphics within the application.

Table 32: Lesson 10 - New Ingots

Adding the AFFilmstrip Ingot

Use the toolbox to draw an AFFilmstrip Ingot on frmTVTPrev. In the Properties window, set properties for the Ingot according to the following table. Use the default setting for all other properties.

Ingot	Property	Setting
	AFFilmstrip	Name flmPrev
	AnimationStyle	1 - Loop
	Height	80
	Interval	200
	Left	22
	Top	76
	Width	119

Table 33: Lesson 10 - New Ingot Properties

Setting the Frames Property

The filmstrip Ingot provides a means of animating a series of graphics. The graphics and their sequence are set through the Frames property of the AFFilmstrip Ingot. To set the Frames property of flmPrev, select Frames property in the flmPrev properties window.

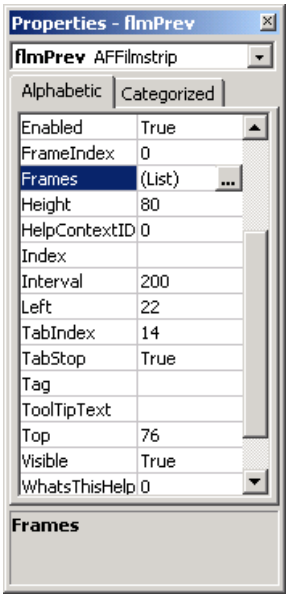

Figure 35: Frames Property in the flmPrev Properties Window

Next, click on the **...** button in the Frames Property. This brings up the frames property window.

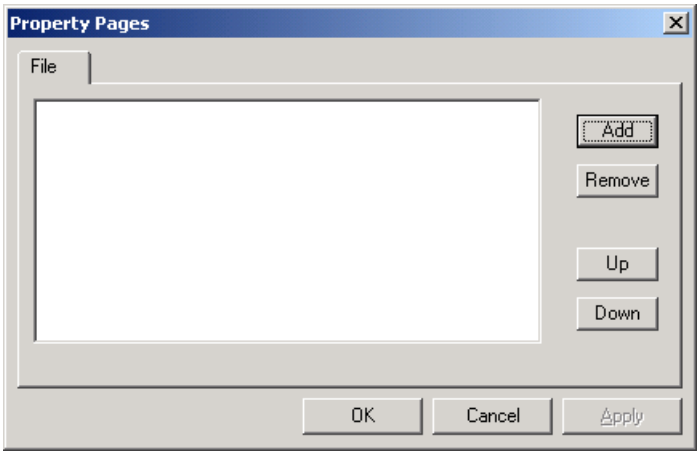

Figure 36: Frames Property Window

Click on the **Add** button to browse for graphics files. Each graphic file you add becomes part of the animation sequence. The sequence of the graphics in the filmstrip animation is changed by selecting a specific graphic and clicking on the **Up** or **Down** button. To remove a graphic from the list, click on the **Remove** button.

The graphics we will use are located in the ...Toolkit\doc\VBTutorial\Graphics folder of the MobileVB™ install

folder. The Property window provides the ability to browse to this location for the files, and will copy the selected files to the current project directory. Alternatively, the files may be manually copied directly from the above folder into the current project directory. The final Frames list should be as follows:

Mov_00.RGX
Mov_01.RGX
Mov_02.RGX
Mov_03.RGX
Mov_03.RGX
Mov_03.RGX
Mov_03.RGX
Mov_03.RGX
Mov_02.RGX
Mov_01.RGX
Mov_00.RGX
Mov_00.RGX
Mov_04.RGX
Mov_05.RGX
Mov_06.RGX
Mov_06.RGX
Mov_06.RGX
Mov_06.RGX
Mov_06.RGX
Mov_06.RGX
Mov_05.RGX
Mov_04.RGX
Mov_00.RGX
Mov_00.RGX
Mov_07.RGX
Mov_08.RGX
Mov_09.RGX
Mov_10.RGX
Mov_11.RGX
Mov_10.RGX
Mov_09.RGX
Mov_10.RGX
Mov_11.RGX
Mov_10.RGX
Mov_09.RGX
Mov_10.RGX
Mov_11.RGX
Mov_10.RGX
Mov_09.RGX
Mov_09.RGX

Mov_09.RGX
Mov_09.RGX

Table 34: Frames Sequence

Make sure to that the Frames list matches the above list exactly. If the order is altered or files omitted the preview will not appear correctly.

Loading the Preview

To load the preview into the Preview form, a new Sub procedure called LoadPreview should be added to frmTVTPrev. The Sub procedure places the category and program name into the user interface and starts the filmstrip preview playing.

Paste the following code into the form module frmTVTPrev:

```
Public Sub LoadPreview(ByVal strProgName As String, _
 ByVal strCatgName As String)
  'loads preview and plays it

  'load the category and program name into the UI
  lblCatgName.Caption = strCatgName
  lblProgName.Caption = strProgName

  'play the filmstrip
  flmPrev.Play
End Sub
```

Calling Sub LoadPreview

The Sub procedure LoadPreview should be called in the Click event code of the Preview button on frmTVTProg. Double-click the Preview button on the Program form to view its Click event procedure. Edit the procedure so that it matches the following (the bold code indicates newly added lines):

```
Private Sub btnPreview_Click()
  'hide this form and show the preview form
  frmTVTProg.Hide
  frmTVTPrev.Show

  'load the preview
  Call frmTVTPrev.LoadPreview( _
   lblProgName.Caption, lblCatgName.Caption)
End Sub
```

Stopping the Filmstrip

Upon exiting the preview screen, the filmstrip needs to be stopped so that it does not continue running. Since there are three buttons on the preview screen that allow users to change screens, each button's Click event must

include code to stop the filmstrip.

In frmTVTPrev, change the grbCatgBack button's Click event to match the following (the bold code indicates newly added lines):

```
Private Sub grbCatgBack_Click()
  'stop the movie
  flmPrev.Stop

  'hide this form and show the category form
  frmTVTPrev.Hide
  frmTVTCatg.Show
End Sub
```

Also, change the Click event for the grbProgBack on frmTVTPrev to match the following (the bold code indicates newly added lines):

```
Private Sub grbProgBack_Click()
  'stop the movie
  flmPrev.Stop

  'hide this form and show the program form
  frmTVTPrev.Hide
  frmTVTProg.Show
End Sub
```

Finally, edit the Click event for the grbTVBack button in frmTVTPrev as follows (new code in bold):

```
Private Sub grbTVBack_Click()
  'stop the movie
  flmPrev.Stop

  'hide this form and show the main form
  frmTVTPrev.Hide
  frmTVTMain.Show
End Sub
```

Running the Project

First, save the changes made to the project and then run the application. Select the Drama category in the AFList-Box. Click on the Show Me button to navigate to the Category form, and move to the 2:00pm timeframe. Select the program "Before The Storm" in the grid. Click on the Show Me button to view the program, and click on the Preview button to move to the Preview form. Once on the Preview form, a filmstrip preview for the TV program should play.

Lesson 11: Uploading the MobileVB™ Project

At this point all the functionality for the TVToday application has been added. This lesson steps through setting project dependencies, compiling, and uploading the TVToday application to a Palm OS® device.

Setting the Dependencies

For TVToday to function properly on a mobile device, we need to specify file dependencies that are referenced only in code (this is, files not used by Ingots such as AFGraphicButton and AFFilmstrip). In the case of TVToday, we need to add Category.PDB and Program.PDB to the list.

MobileVB™ project dependencies are set through the Project Properties Dialog. Open the Project Properties by selecting the **MobileVB™ Project Settings...** option from the MobileVB Menu.

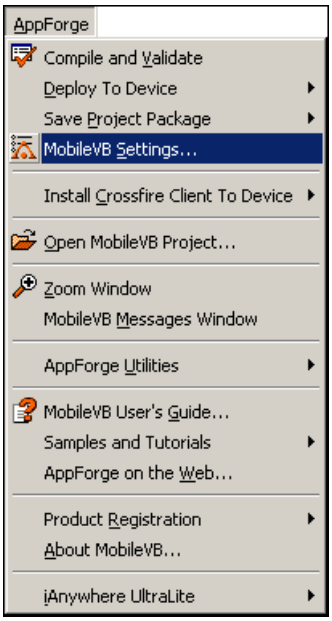

Figure 37: MobileVB Menu with MobileVB Settings Selected

This will launch the MobileVB Settings window. Click on the Dependencies tab and the dependencies, currently blank, will appear.

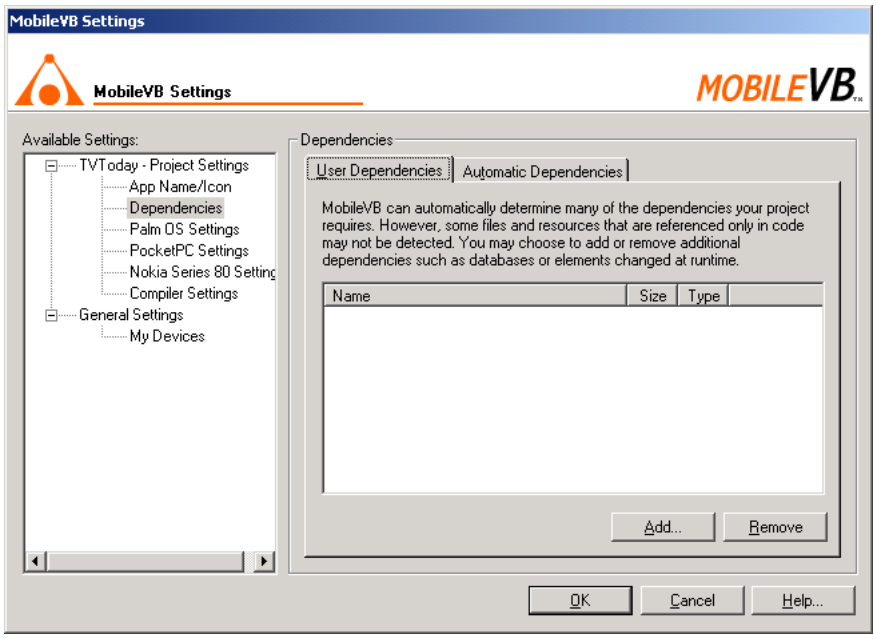

Figure 38: Blank Dependencies Page

Click on the **Add** button, and add the following files to the list:

Dependency Files
Category.PDB
Program.PDB

Table 35: Dependencies

Click OK to exit the MobileVB Settings window.

Uploading the Project to a Mobile Device

NOTE: If the target device in running Pocket PC, make sure it is hooked up to the desktop computer and that the device's ActiveSync® connection is activated.

To upload TVToday to a mobile device, begin by selecting **Deploy to Device** from the MobileVB Menu.

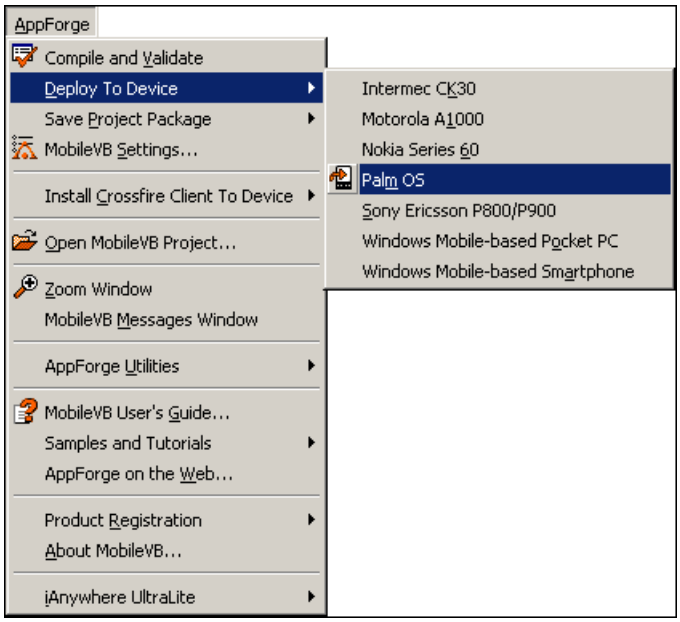

Figure 39: Deploying the Project

If the TVToday project was not saved prior to the start of the upload process, the dialog box prompts you to save your project before compiling. Click Yes to save and continue the uploading process.

Figure 40: Click Yes to Continue Upload

Now MobileVB™ will compile the TVToday application. A compilation progress box should appear.

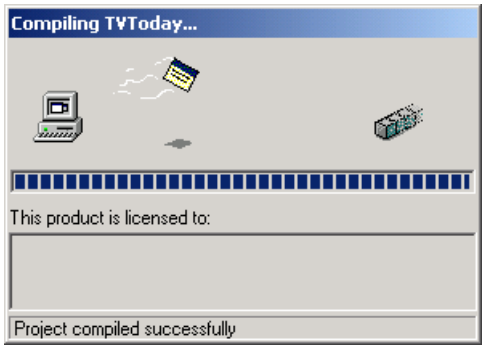

Figure 41: Compilation Progress Box

When compiling the code in a project, MobileVB analyzes the code for possible errors or conflicts that may

prevent a project from functioning properly once uploaded to the target hardware. If MobileVB™ finds any potential difficulties, it will identify them. Otherwise, the code will be compiled and the upload will continue.

If the target hardware is a Palm OS device, following the compilation of TVToday, MobileVB transfers the program to the device's Install folder. The following sequence of windows should appear:

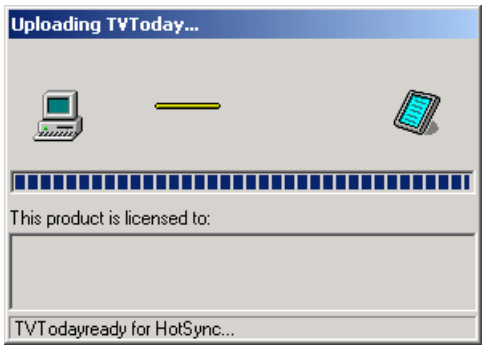

Figure 42: Transfer In Progress

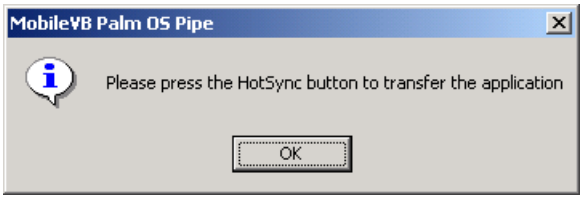

Figure 43: HotSync Notification

Click **OK** to finish the upload process.

If the target device is running Palm OS®, initiate a HotSync™ operation. Once TVToday and its databases have been uploaded to the handheld device, the TVToday Icon should appear in the device's applications menu. Tap on it to start TVToday.

If the target hardware is a Pocket PC device, MobileVB™ transfers the program to the device via the ActiveSync® connection. It will be installed automatically in the My Device\Program Files\AppForge Projects folder. Tap on the TVToday entry in this directory to launch the application.

Viewing Messages in the Compiler

If errors are discovered in the process of compiling a MobileVB™ project, the progress window will indicate there is an error. Additionally, MobileVB provides a Compiler Messages window that displays the potential error.

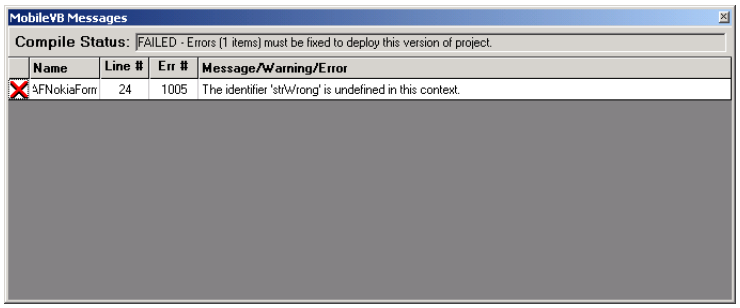

Figure 44: MobileVB(tm) Compiler Messages

Congratulations!

You have successfully developed the TVToday application and completed the MobileVB™ tutorial.