www.realbasic.com

Tutorial

REAL Studio 2010 Release 1 Tutorial

Documentation by David Brandt.
Concept by Geoff Perlman.

Printed in U.S.A.

Mailing Address	REAL Software, Inc. PO Box 162181 Austin, TX 78716
Web Site	http://www.realsoftware.com
ftp Site	ftp://ftp.realsoftware.com
Support	Submit via REALbasic Feedback at www.realsoftware.com
Bugs/Feature Requests	Submit via REALbasic Feedback at www.realsoftware.com.
Sales	sales@realsoftware.com
Phone	512-328-REAL (7325)
Fax	512-328-7372

Version 2010 Release 1, February, 2010.

REAL Studio Tutorial

Welcome to the REAL Studio Tutorial!

This *Tutorial* is for people who are new to programming and new to REAL Studio. It will give you an introduction to the REAL Studio development environment, lead you through the development of a real application, and show you what kinds of other applications can be built with REAL Studio.

The *Tutorial* assumes that you have already completed the *Quickstart*. It should take you about an hour to complete this *Tutorial*.

Note If you have experience with other versions of the BASIC language or other programming languages, you'll want to check out the *User's Guide*.

Presentation Conventions

The *Tutorial* uses screen snapshots taken from the Windows, Macintosh, and Linux versions of REAL Studio. The interface design and feature set are identical on all platforms, so the differences between platforms are cosmetic and have to do with the differences between the Vista Aero interface, the Mac OS X Leopard interface, and the Linux Ubuntu distribution.

Italic type is used to emphasize the first time a new term is used and to highlight important concepts. In addition, titles of books, such as *REAL Studio User's Guide*, are italicized.

When you are instructed to choose an item from one of REAL Studio's menus, you will see something like "choose File ▶ New Project". This is equivalent to "choose New Project from the File menu."

The items within the parentheses are *keyboard shortcuts* and consist of a sequence of keys that should be pressed in the order they are listed. On Windows and Linux, the Ctrl key is the modifier; on Macintosh, the ⌘ (Command) key is the modifier. For example, when you see the shortcut "Ctrl+O or "⌘-O", it means to hold down the Control key on a Windows or Linux computer and then press the "O" key or hold down the ⌘ key on Macintosh and the press the "O" key. You release the modifier key only after you press the shortcut key.

Bold is used to indicate text that you will type while using REAL Studio.

Some steps ask you to enter lines of code into the REAL Studio Code Editor. They appear in Frutiger (a sans serif font) in a shaded box:

```
ShowURL SelectedURL.Text
```

When you enter code, please observe these guidelines:

- Type each printed line on a separate line in the Code Editor. Don't try to fit two or more printed lines into the same line or split a long line into two or more lines.
- Don't add extra spaces where no spaces are indicated in the printed code.

Whenever you run your application, REAL Studio first checks your code for spelling and syntax errors. If this checking turns up an error, REAL Studio will direct your attention to the line of code that is causing problems. Check the line against the printed line. The *Tutorial* includes troubleshooting sections that help you handle syntax errors.

Getting Started

You'll build an application that manages URLs and email addresses. A URL (which stands for *Uniform Resource Locator*) is the address of a web page that you type into the Address area in your web browser. This application will launch your default web browser application and display the web page that you entered.

Figure 1 shows a URL in an internet browser's Address area:

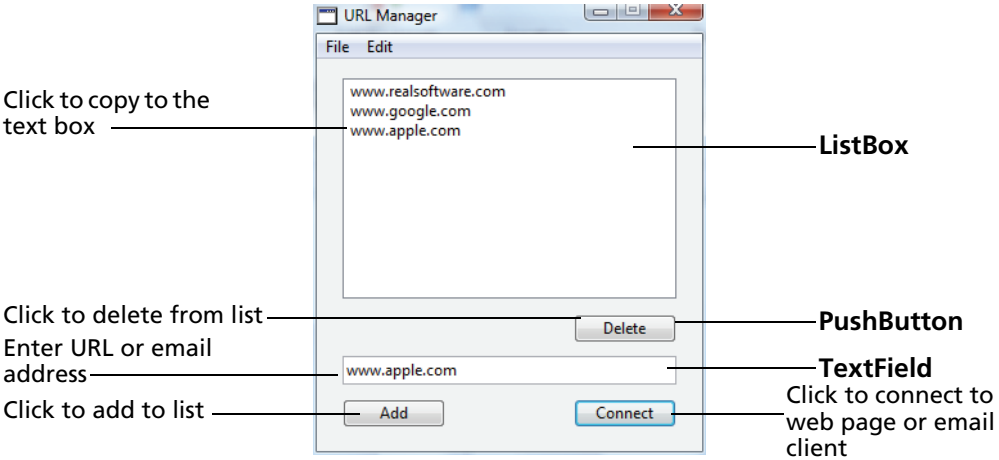
Figure 1. A URL entered into a browser.

If you enter an email address instead of a URL, the application will launch your email application, create a new blank email, and enter the address into the "To" box. Figure 2 shows how to enter an email address into a browser's Address area.

Figure 2. An email address entered into a browser's Address area.

When you are finished, the application will look like Figure 3. In Figure 3, the labels in bold indicate the types of controls that were used in creating the application. The functions of the controls are described in plain type.

Figure 3. The finished URL Manager application.



You use the TextField to enter the URL or email address you want and then click the Connect button. To save it in the list, you click the Add button. To select a URL in the ListBox, highlight it in the list and then click Connect. If you don't need the URL any more, highlight it in the list and then click the Delete button.

The application uses three types REAL Studio controls to do most of the work for you:

- A **ListBox** is the type of control that holds scrollable lists. It can hold both single- and multiple-column lists and scroll horizontally and vertically.
- A **TextField** is the type of control that holds a single line of text.
- A **PushButton** is a standard pushbutton. It is most often used to initiate an action (as it is here).

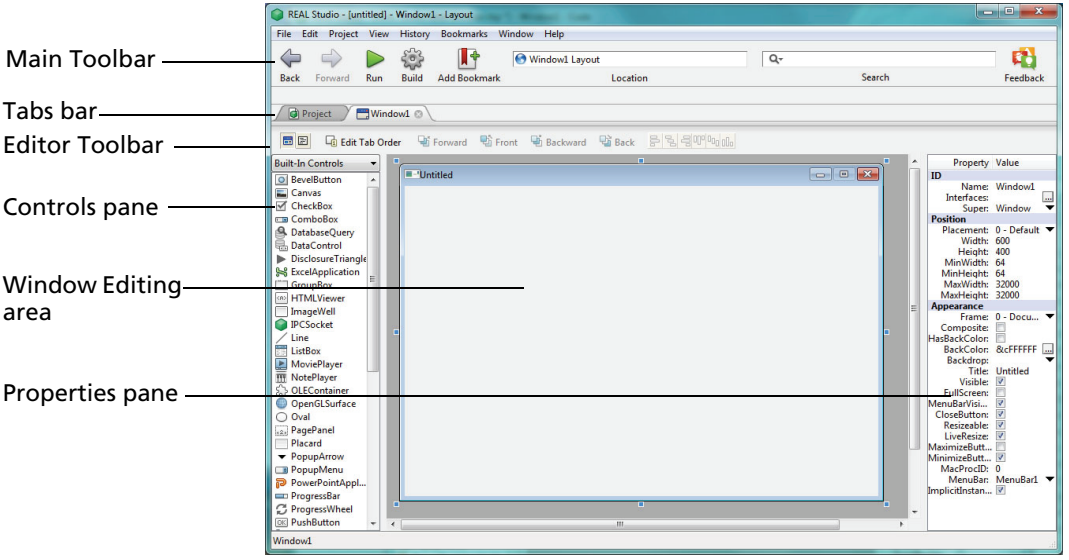
Creating the Interface

If you have not already done so, double-click the REAL Studio application icon. In a few moments, the REAL Studio Integrated Development Environment window (IDE) appears. If the Welcome screen appears, close it to get to the REAL Studio IDE.

The IDE uses a tabbed interface to display multiple editors in the same window. REAL Studio displays the Window Layout Editor as the top panel in the IDE. You switch among the editors by clicking on the Tabs bar. Notice that a second tab for the Project Editor is already open. We will get to the Project Editor later. Your first task is to build the interface with the Window Layout Editor.

The editor for Window1 is shown in Figure 4.

Figure 4. The Window Layout Editor for Window1.



The Window Layout Editor is divided into three panes. The pane on the right is the Properties pane and it shows the properties of the selected item in the Window Editing area. In Figure 4, the window is selected, so the Properties pane shows the properties of the window.

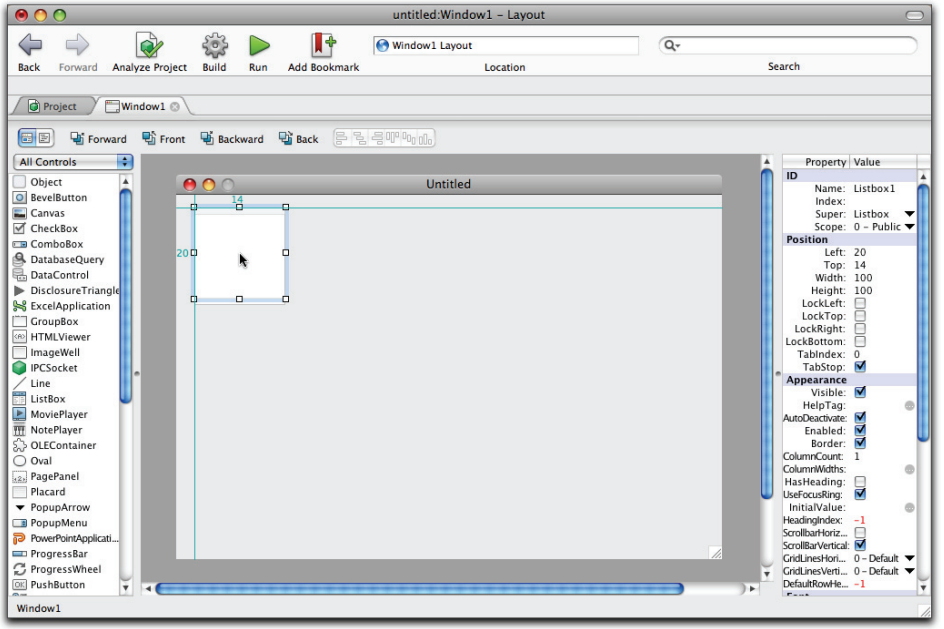
The center pane is the Window Editing area. It is where you design the window. In Figure 4, the window is blank because you haven't added any interface items yet. It shows only a default document window.

The pane on the left is the list of controls that you can add to a window. Above the list is a pop-up menu that controls the types of controls the list displays. When set to "Built-in Controls" it shows the complete list of built-in REAL Studio interface elements. It contains items for the ListBox, TextField, PushButton, and many more. You add a control to a window by dragging it from the list to a location in the window.

1 In the Controls pane, hold down the mouse button on the ListBox item and drag it to the top-left corner of the window in the Window Editing area.

As you drag toward the corner, alignment guides will appear, as shown in Figure 5. You may need to drop the ListBox near the correct area and then drag it again to see the alignment guides.

Figure 5. Alignment guides help you position the ListBox and align other controls.

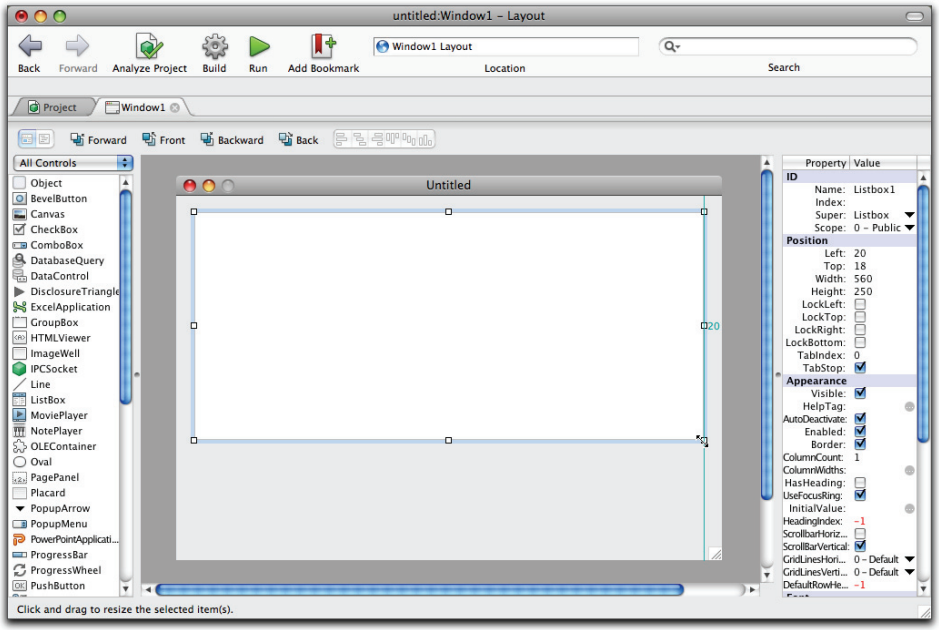


2 Release the mouse button.

A square ListBox appears in the Window Editor.

3 Grab the resizing handle in the bottom-right corner of the ListBox and drag it diagonally to enlarge the ListBox. When the alignment guide appears on the right, release the mouse button, as shown in Figure 6.

Figure 6. Enlarging the ListBox.

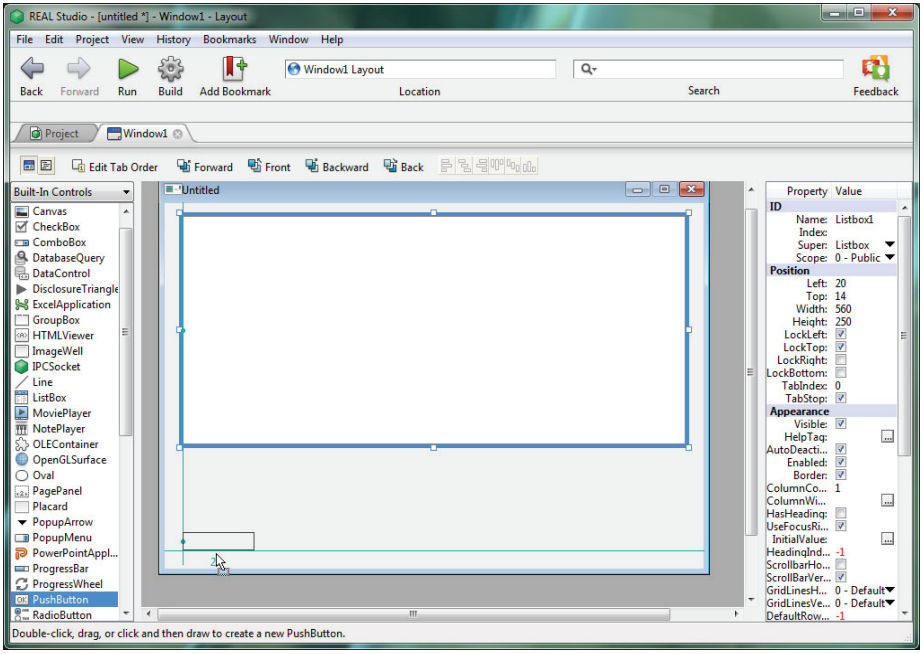


4 Center it in the upper area of the window, as in Figure 6.

5 Next, drag a PushButton control from the Controls pane to the lower-left corner of the window (where the "Add" button is in Figure 3 on page 5).

As you drag toward the left edge of the ListBox, a vertical alignment guide appears. Use the vertical alignment guide to position the PushButton, as shown in Figure 7.

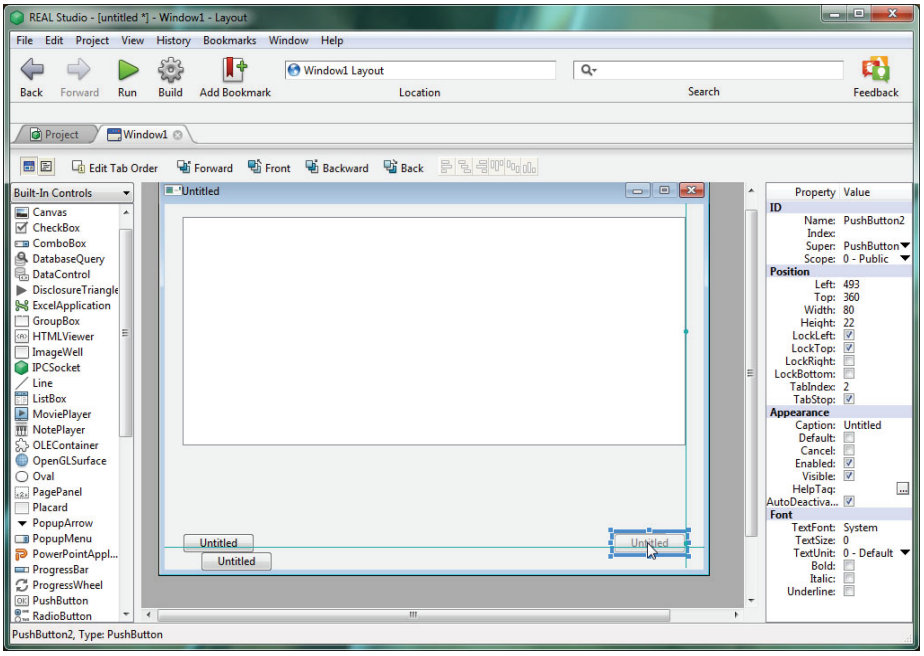
Figure 7. Aligning the Add PushButton.



6 With the PushButton selected, press Ctrl+D (Windows and Linux) or ⌘-D (Macintosh) or choose Edit ▶ Duplicate to make a copy of the PushButton. Move it to the right of the previous button and align it with the right edge of the ListBox.

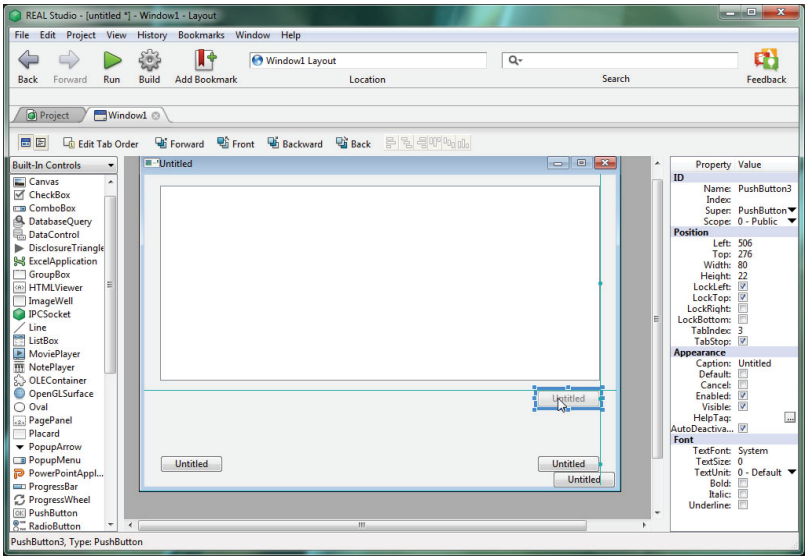
When you reach the approximate position, both vertical and horizontal alignment guides appear. Align the baselines of the two PushButtons' captions, as shown in Figure 8.

Figure 8. Aligning the Add and Connect PushButtons.



7 With this PushButton selected, press Ctrl+D (Windows and Linux) or ⌘-D (Macintosh) or choose Edit ▶ Duplicate to make a copy of the PushButton and drag the PushButton upward vertically, just below the ListBox. Release the mouse button when the horizontal alignment guide appears.

Figure 9. Adding the Delete button.

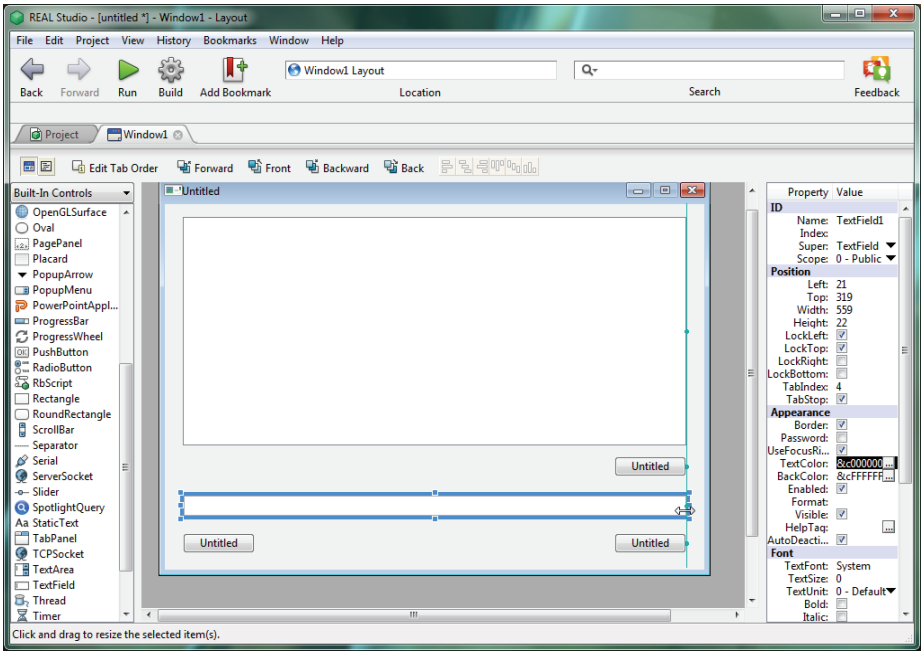


8 In the Controls pane, hold down the mouse button on the TextField and drag it into position in the window, midway between the Delete and Connect buttons (see Figure 10 on page 11).

9 Align the left edge of the TextField with the left edge of the ListBox.

10 Use the center selection handle to stretch the TextField so that its right edge aligns with the right edge of the ListBox.

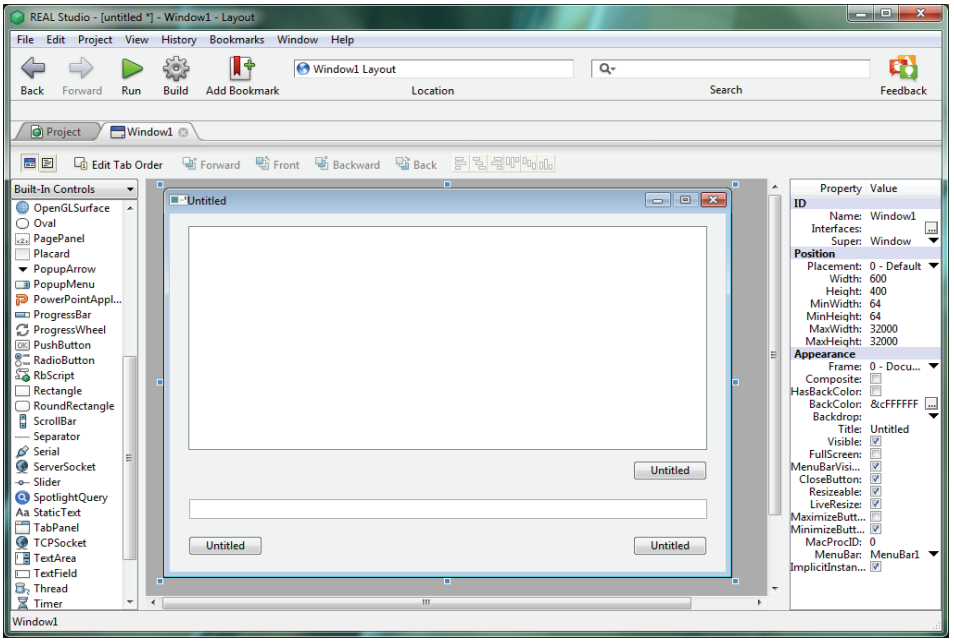
Figure 10. If Aligning the right side of the TextField with the ListBox and Pushbuttons.



If you find that the top PushButton, the TextField, and the bottom row of PushButtons are not evenly distributed, you can reposition a control by clicking on it and moving it one pixel at a time using the Up or Down arrow keys.

Your application's interface is now complete! It should look like Figure 11.

Figure 11. The finished interface in the REAL Studio IDE.



Before going any further, save the project.

- Choose File ▶ Save (Ctrl+S or ⌘-S) and name it **URL Manager**.

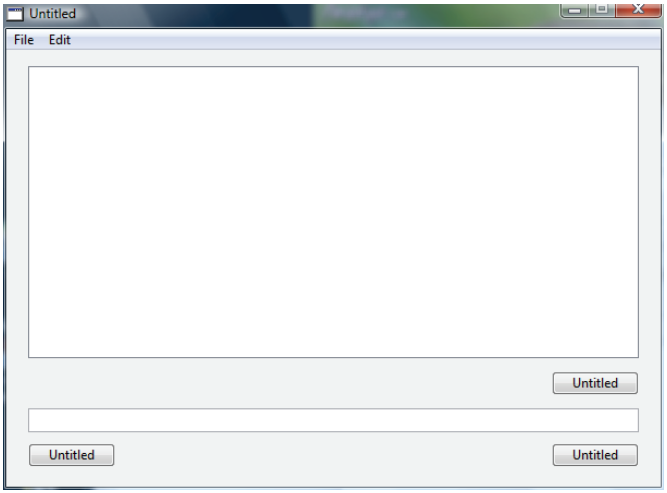
At this point, you can actually try it out. Of course it won't do anything since we haven't told any of the interface elements what to do.

1 Just for fun, click the Run button in the Main Toolbar.

REAL Studio builds the application and opens it in its own window.

The first version of the application should look like this:

Figure 12. The first version of the application.



In this state, the PushButtons work — that is, you can click them and they highlight — but they don't do anything because we haven't told the PushButtons what to do when they're clicked. You can enter text into the TextField — but it doesn't go anywhere because there are no instructions to process this text. And the ListBox is all set to display and scroll items but we don't have any way to get text into the ListBox yet.

Notice that the application has File and Edit menus. On Windows and Linux, the File menu has one item, Exit, that quits the application and returns to the REAL Studio IDE. On Mac OS X, the Quit menu item appears in the application's menu instead of the File menu.

The Edit menu has the standard Undo, Cut, Copy, Paste, Delete, and Select All menu items. Initially these items are all disabled, but you can enable them by entering and selecting text in the TextField.

2. Type some text in the TextField and then pull down the Edit menu.
   Notice that the Select All menu item is enabled.
3. Choose Edit ▶ Select All and then choose Edit ▶ Cut.
   When you cut text to the Clipboard, the Paste item becomes enabled. All of the standard Edit menu items work on text without any programming on your part.

   Now we need to go back to the REAL Studio IDE to make this application operational.
4. In the URL Manager application, choose File ▶ Exit on Windows or Linux (or My Application.debug ▶ Quit on Mac OS X) to return to the REAL Studio IDE. On Windows and Linux, you can also quit the application by clicking the window's Close box.

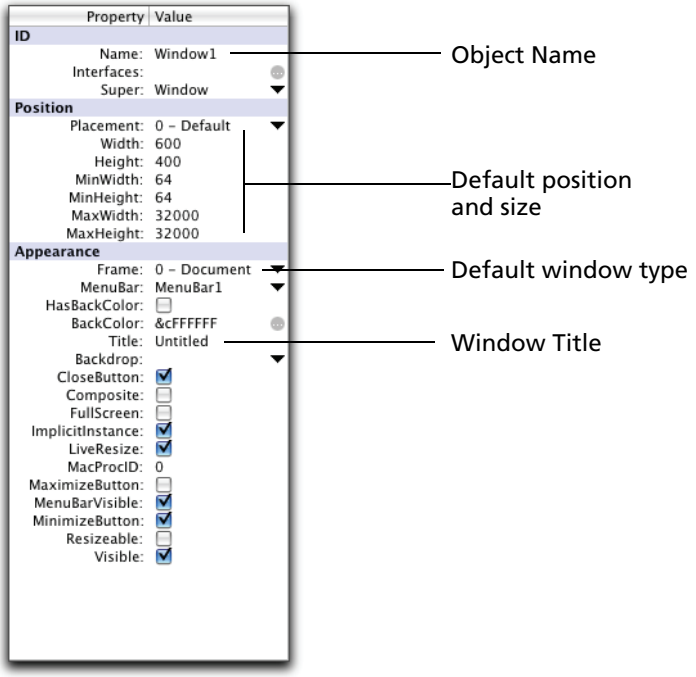
Giving Objects Meaningful Names and Labels

You've already noticed that all the PushButtons use the default label "Untitled." They also have default names like PushButton1, PushButton2, etc. Before getting too far into the project, we should give the objects meaningful names and labels. You refer to the object's names in your REAL Studio code and, of course, the labels are presented to the user. You should give each object that you refer to in your code a meaningful name at the start of the project.

1 Click on the surface of the window in the Window Editor — not one of the controls in the window — and then take a look at the Properties pane.

The window has the default name "Window1", which is shown in the first line. If it shows some other name, then you clicked on one of the controls in the window rather than the surface of the window itself.

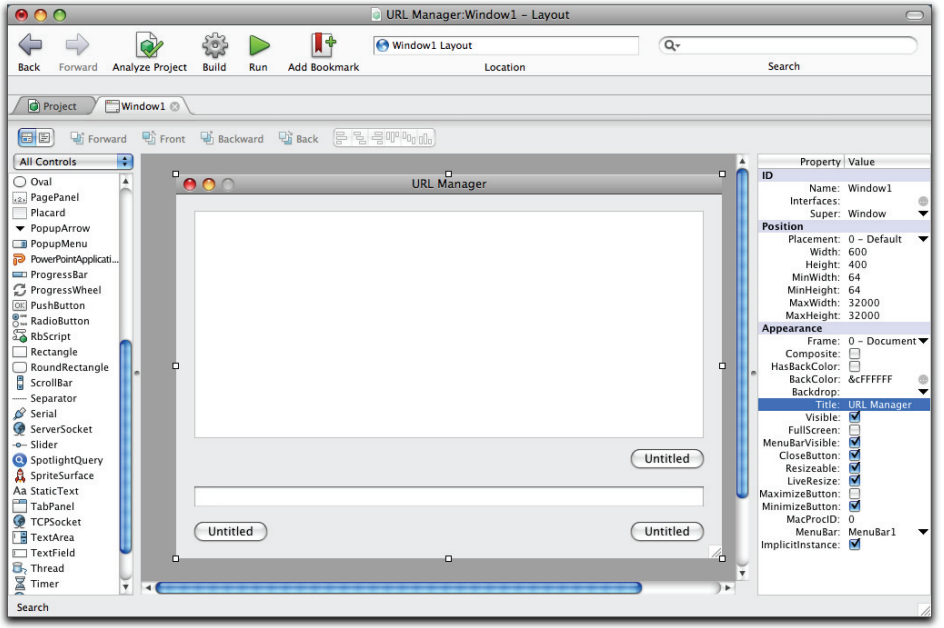
Figure 13. Window1's Properties pane.



The text that appears in the Title bar of the window is the Title property. The Name property is the name of the window that you use in your code to refer to the window.

2 Select the default Title, "Untitled", in the Properties pane and replace it with the text **URL Manager**, and press the Enter key (Return on Macintosh). When you press Enter or Return, the new title appears in the Title bar of the Window Editor as well as in the Properties pane.

Figure 14. Changing Window1's Title property.



Similarly, we need to replace the default names and labels for the controls in the window.

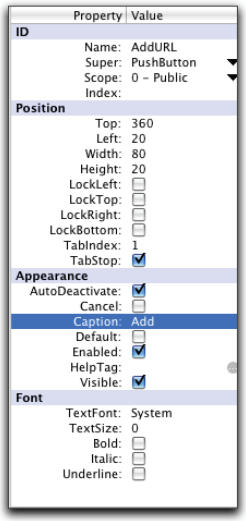
3 In the Window Editor, select the Untitled button in the lower left corner by clicking on it.

Notice that the Properties pane changes to show the properties of this PushButton. The three PushButtons are named PushButton1, PushButton2, and PushButton3. They were named in the order they were created. We'll never remember which one is which, so it's best to rename them at the same time we're entering their labels.

4 Change this PushButton's Name property to **AddURL** and its Caption property to **Add**. Press Enter (or Return on Macintosh) after typing each entry to save each new property value.

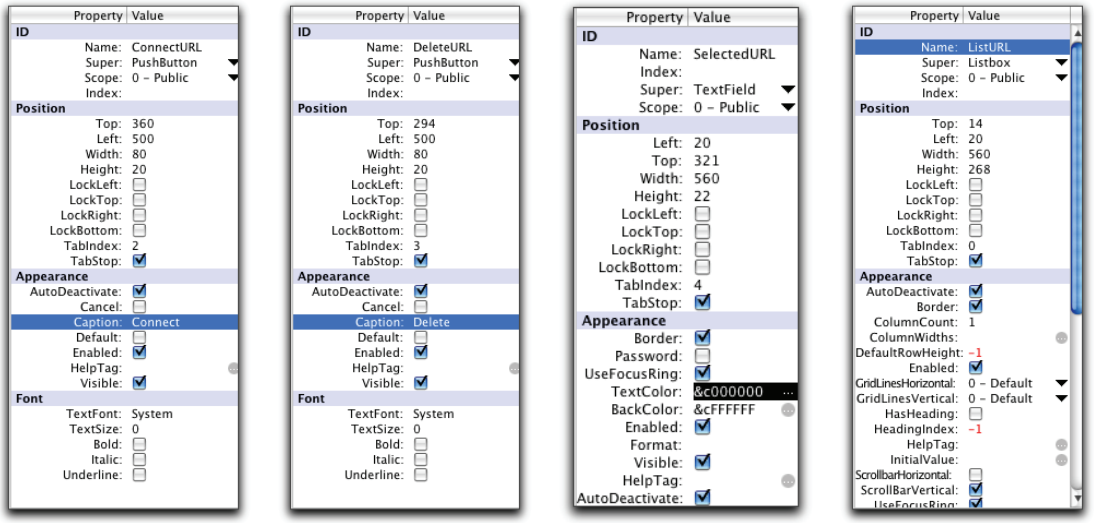
Notice that the Caption text immediately replaces "Untitled" in the Window Editor when you press Enter. When you are finished, the Properties pane for the AddURL PushButton should look like this.

Figure 15. The Properties pane for the Add button.



5. Click on the Untitled button in the lower right to select it. It's PushButton2. Use the Properties pane to change its name to **ConnectURL** and its Caption property to **Connect**.
6. Click on the Untitled PushButton between the ListBox and the TextField. Change its Name property to **DeleteURL** and its Caption property to **Delete**.
7. Click on the TextField and change its name to **SelectedURL**.
8. Click on the ListBox and change its Name property to **ListURL**.
   That takes care of it. The Properties panes for these objects should look as shown in Figure 16.

Figure 16. The Properties panes for the other controls.



Connect button | Delete button | TextField | ListBox

Check your work to be sure that the items are named correctly. If there is a spelling error in a Name property, the code that is supposed to refer to that item will not work.

In the Window Editor the three buttons should now look like this:

Figure 17. The three PushButton controls after renaming.

9 Choose File ▶ Save to save your changes.

10 Click the Run button in the Main Toolbar to test the application.
It doesn't do any more than the last version, but at least all the interface elements are labeled correctly.

11 When you are finished testing the application, choose File ▶ Exit (on Windows and Linux) or My Application.debug ▶ Quit (on Mac OS X) to return to the REAL Studio IDE. On Windows and Linux, you can also quit out of the test application by clicking the window's Close box.

Be sure to quit out of the built application before resuming your work in the REAL Studio IDE. If you don't, close the "Run" tab in the Tabs bar.

Making the URL Manager Do Something

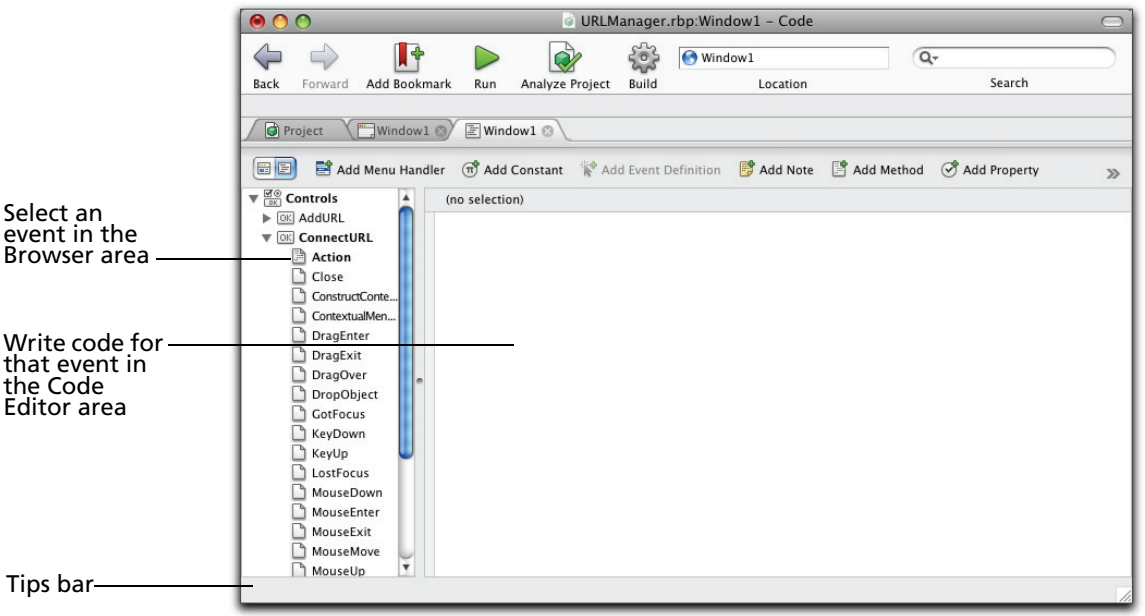
Now that the interface is designed and its appearance is touched up, it's time to make the controls do their jobs. We'll start with the Connect button. You specify the action that a button carries out by writing some code that becomes a part of the button.

1 In the Window Editor, double-click the Connect button.
The Code Editor for Window1 appears. This is where you write code for the window and the controls contained in the window.

On the left side is a browser area that lists all the controls that we've added to the window, among other things. (For the *Tutorial*, we only need to work with the controls.) On the right side is the code editing area. It holds the code for the item that is selected in the browser.

Right now the "Action" item for the Connect button is selected. It's highlighted in the Browser area and the header area above the editor gives the name of the event.

Figure 18. The Code Editor for Window1.



In order to get the Connect button to do something, we need to write some code that will run only when the user clicks this button. Fortunately, the REAL Studio application itself monitors all user interface activity while the application is running and it knows whenever this happens.

To make the Connect button work, we need to write the instruction that connects the user to the web site that the user enters into the TextField.

In the Browser area of the Code Editor, you'll see a list of events that REAL Studio continuously monitors while the application is running. For example, the MouseEnter and MouseExit events are triggered when the mouse pointer enters the region of the PushButton and leaves that region.

The event we need is the "Action" event. This event takes place when the user clicks the button.

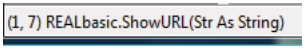
On the right side of the divider, you can write the code that runs automatically when the user clicks the button. (Notice that the top line, "Sub Action()", indicates which event the code is for). The instruction to open a web site in the user's browser (or open the email application) is simple. The instruction is ShowURL and its syntax is:

```
ShowURL string
```

Where *string* is the URL (or email address).

As soon as you enter "ShowURL", the syntax for this command appears in the Tips bar. By referring to the Tips bar you can avoid having to look up a command's syntax in the Online Language Reference whenever you need help.

Figure 19. The Tips bar showing the syntax for ShowURL.

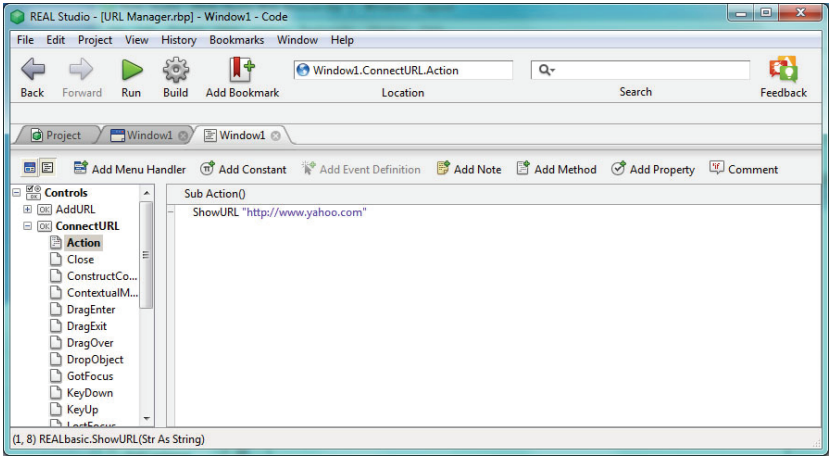


2 To test out this button, enter the following line in ConnectURL's Action event:

```
ShowURL "http://www.yahoo.com"
```

Be sure to enclose the URL in quote marks. The quote marks indicate that the text in quotes is a string (rather than a number or some other type of data). Your Code Editor should now look like this:

Figure 20. The code for ConnectURL's Action event.



3 Click the Run button in the Main Toolbar to test the Connect button.

4 When the application window appears, click the Connect button.
In a few moments, your default web browser will launch and bring up the web page you entered. (This, of course, assumes your computer has a connection to the Internet and you have a default web browser application.)

Figure 21. The Yahoo home page.

Of course, we need to modify the code so that the text passed to the ShowURL command can be entered by the user while the application is running. When we use ShowURL "http://www.yahoo.com", the particular URL is "hardcoded."

Note If you get a syntax error, check to make sure that you spelled the ShowURL command correctly, that you enclosed the URL in double quote marks, and that it is a valid URL.

5 Select the URL Manager application window and choose File ▶ Exit on Windows or Linux (or My Application.debug ▶ Quit on Mac OS X) to return to the REAL Studio IDE.

We now need to replace the text used with the ShowURL command with the code that refers to the contents of the TextField, SelectedURL.

Since the TextField is named "SelectedURL", you might think that we could write:

```
ShowURL SelectedURL
```

but that won't work because "SelectedURL" is the name of the object itself. It has lots of properties — like its position in the window, whether it takes several lines of text or just one, whether it can accept styled text, and so forth. If you use "ShowURL SelectedURL", REAL Studio would have no idea what you mean. Besides, the ShowURL command only accepts a text string, not a control.

When you need to refer to one of an object's properties, you write the name of the object, followed by a dot, followed by the name of the property. In other words, you use this syntax:

```
objectname.propertyname
```

It's sometimes called "dot" notation.

In this case, the TextField is named "SelectedURL" and the TextField property that we want is its "Text" property. This means the following expression accesses the contents of the TextField:

```
SelectedURL.Text
```

That is, "SelectedURL" is the name of the object and "Text" is the name of the object's property that we need.

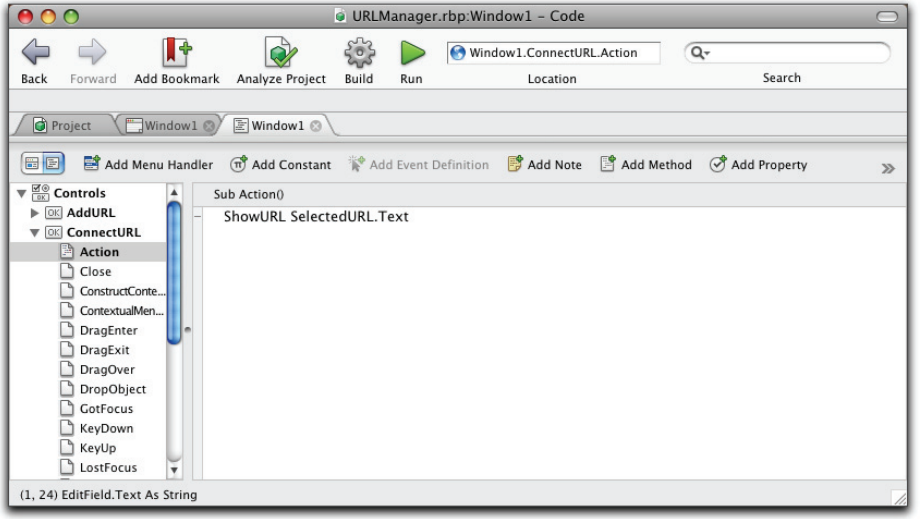
1 In the Code Editor for the Action event, modify the code to read:

```
ShowURL SelectedURL.Text
```

This expression SelectedURL.Text refers to the text property only.

Your Code Editor should now look like this:

Figure 22. The revised code for the Action event.



If you have trouble entering this line of text, be sure you have quit out of the test application before returning to the REAL Studio IDE. Simply quit the test application.

2 Save the project (File ▶ Save) and test it by clicking the Run button in the Main Toolbar.

The URL Manager application opens in a new window.

3 Enter a URL in the TextField, such as **http://www.realsoftware.com** and click Connect.

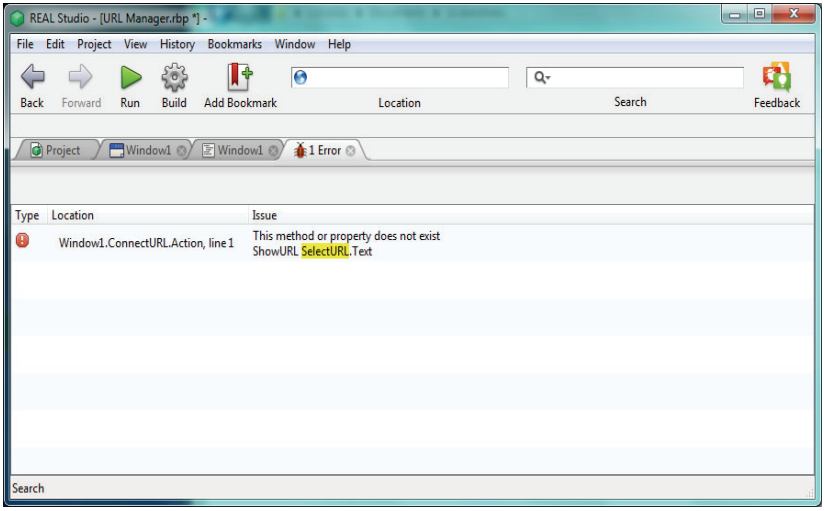
Your default web browser should launch and open the web page you entered.

4 When you're finished, quit out of the debugging environment (File ▶ Exit on Windows and Linux (MyApplication.debug ▶ Quit on Macintosh) to return to the REAL Studio IDE.

If the Application Doesn't Run

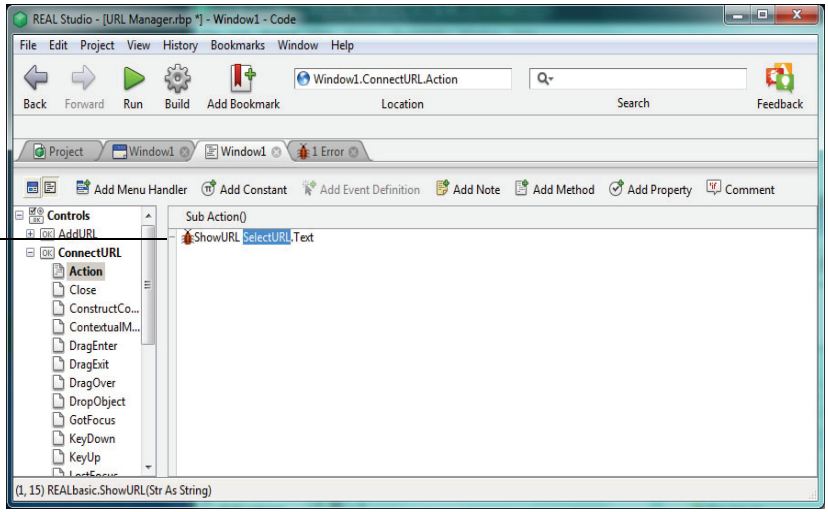
If REAL Studio was unable to launch the debugging application, it's because it couldn't recognize a term you entered into the Code Editor. For example, if you misspelled either "ShowURL" or "SelectedURL," REAL Studio stops and points out the term it doesn't recognize. Instead of compiling and launching the application, it opens a new "Issues" tab in the IDE. It has one line per issue. For example, the following shows a spelling error.

Figure 23. A spelling error results in an unrecognized term.



The misspelled term is highlighted. Double-click the line to go to the editor with the error. In this case, REAL Studio displays the Code Editor for the ConnectURL Action event, shown in Figure 24. The line that contains the error has a bug icon to its left.

Figure 24. A misspelled object name.



Since it can't find an object called "SelectURL," it can't create the application for you. REAL Studio knows it would never be able to figure out what to do when a user clicks the Connect button. Be sure you've renamed the controls as described and referred to their correct names in the Code Editor. You'll also get this error if you wrote ShowURL SelectedURL and left off the name of the property that contains the contents of the TextField.

Another possibility is that you used the correct name in the Code Editor but did not rename the TextField using the Properties pane as shown in Figure 16 on page 17. In that case, the statement in the Code Editor as shown in Figure 22 ought to work, but there is no object named "SelectedURL" in the application. If you get an error message, start by checking the highlighted term.

Now, we'll make the other controls do their jobs.

The Add Button

The Add button is supposed to take the text in the TextField and add it to the end of the list in the ListBox. That's easy.

1. If the Code Editor for the window is not already open, double-click the Add button in the window (If the Window Editor is not shown, click the Window1 tab in the Tabs bar).
2. Enter the following code into the Add button's Action event:

```
ListURL.AddRow SelectedURL.Text
```

The first part of the expression, ListURL.Addrow calls a built-in method belonging to a ListBox. The AddRow method is a command that adds a row of text to the end of whatever list is already in the ListBox. Not surprisingly, it needs to be passed the

text of the new item. We already know that SelectedURL.Text refers to the contents of the TextField, so that is what we use.

Note A *method* is a command that performs an action. Technically, ShowURL is a *global method* because it isn't attached to any particular object. It can be called by any object that can call a method. We just happen to be calling it from a PushButton. (We could, for example, design the application so that ShowURL is called when the user chooses a menu item instead of clicking a button.)

Just as objects can have properties (like their name, size, position, and label), they can also have their own methods. AddRow is also a method but it "belongs" only to ListBoxes. It has a specialized action that only makes sense when applied to lists in ListBoxes.

The Delete Button

The Delete button removes the selected item in the ListBox. It's also pretty simple.

1. In the Code Editor, expand the DeleteURL item and highlight the Action item.
2. Enter the following line of code:

```
ListURL.RemoveRow ListURL.ListIndex
```

In this case, we are calling the built-in RemoveRow method of the ListBox control. Instead of text, it needs the number of the row (line) to delete. The ListIndex property contains that number, so we pass that number to the RemoveRow method.

Note If you place the insertion point in the RemoveRow term, the Tips bar will say that RemoveRow requires that an integer be passed to it.

The ListBox

The ListBox itself has the job of copying the item the user selects into the TextField so the user can connect to that URL or email address. You can easily do this by writing an *assignment statement* that runs when the user highlights an item in the list.

1. Expand the ListURL item in the Code Editor browser area and select the Change event handler.
   The Change event runs whenever a different item in the ListBox is highlighted.
2. Enter the following code into the Change event:

```
SelectedURL.Text=ListURL.Text
```

The ListBox's Text property contains the text of the highlighted item. The Text property of SelectedURL contains the text that's currently displayed. This assignment statement copies this text into the Text property of the TextField. The Text property stores the text that is displayed in the TextField.

Testing the Application

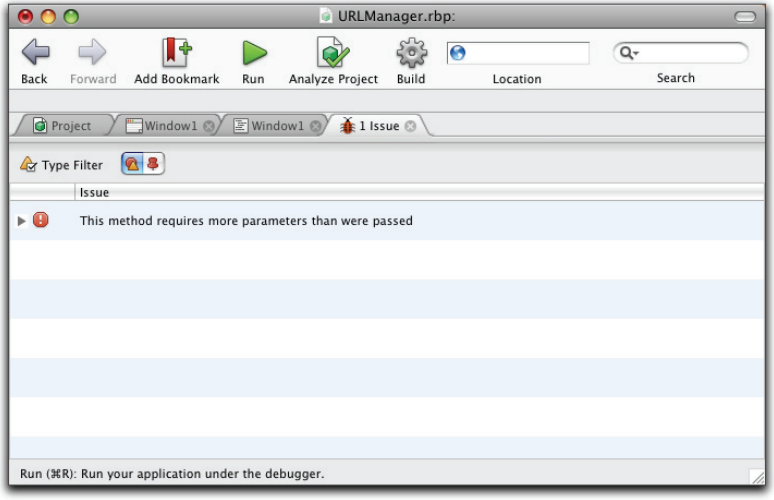
That's the basics of the application. Now it's time to test out all these features. Before you run your project again, you can ask REAL Studio to check the project for errors or ways to optimize your application. You use the Analyze Project command. Analyze Project uses REAL Studio's compiler to review all of your code and identify any coding errors. It also offers suggestions on how to improve your application.

1 Choose Project ▶ Analyze Project.

If it finds no issues, it displays the message, "Check passed; no issues were found" in the Tips bar and does *not* open an "Issues" panel.

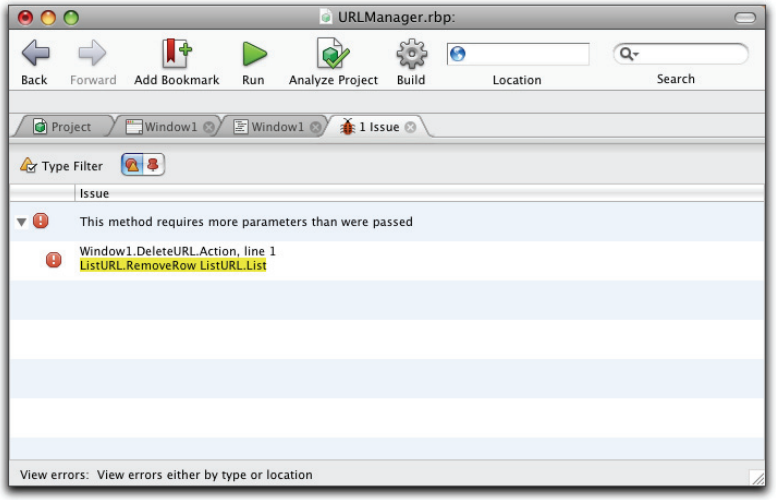
If Analyze Project finds any issues, it opens up an "Issues" panel and lists the problems. An example of an error that prevents the compilation of the project is shown below:

Figure 25. An error discovered by the Analyze Project command.



If there are several issues, they are organized by type of issue. Click the disclosure triangle to the left of the error message to view the specific errors.

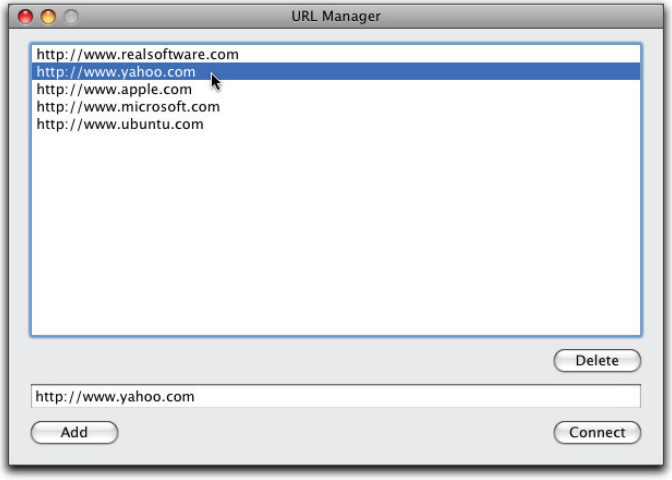
Figure 26. Details of the error shown in Figure 25.



2 (If needed) Double-click each error message and fix the error before proceeding.

3 Click the Run button in the Main Toolbar to test the application.
The application looks the same as it did, but all the controls work! For example, enter **http://www.realsoftware.com** into the TextField and click the Add button. Add a few other URLs to the ListBox in this manner and then highlight one in the ListBox to move it to the TextField.

Figure 27. Clicking a URL to move it to the TextField.



You can:

- Enter a URL into the TextField and connect to the site using your default web browser.
- Add the URL to the ListBox.
- Select any URL in the ListBox to copy it into the TextField.
- Delete the selected URL in the ListBox.

If you want to send an email, enter it in the following way:

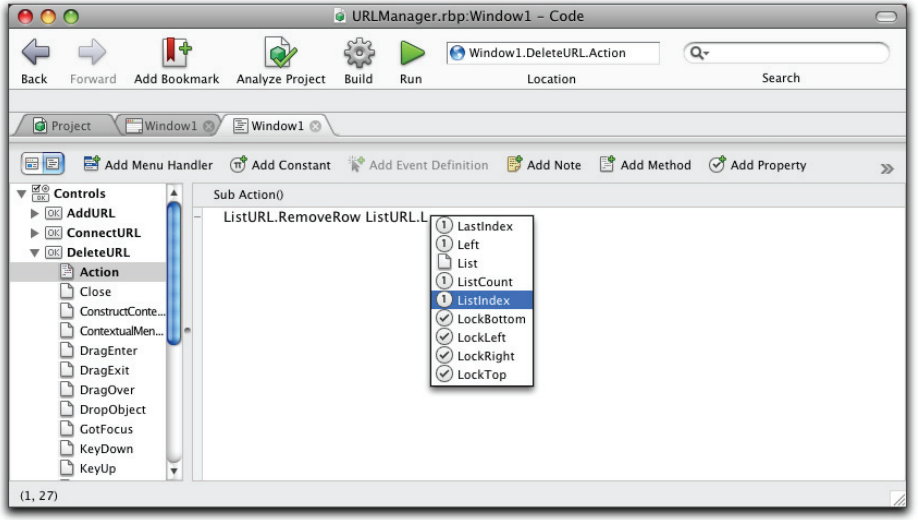
```
mailto: username@domain.com
```

Using Autocomplete

One way to avoid using incorrect terms is to take advantage of REAL Studio's "autocomplete" feature. As you type, REAL Studio tries to guess what you are typing. If you type the first few characters of a REAL Studio language object — either built-in or a variable, method, or property that you created — it shows its guess in light gray type. If the guess is correct, press the Tab key to complete the entry.

If there is more than one possibility, REAL Studio displays a contextual menu when you press Tab. Here is an example:

Figure 28. Autocomplete in action.



The user has just typed "ListURL.L" and pressed the Tab key. REAL Studio displays a contextual menu with all the possible terms that complete the expression. Use the Up and Down arrow keys to navigate through the list and then press the Enter key (Return on Macintosh) to select the highlighted term.

The small icons to the left of the term tell you what type of object it is. In this case, the checkmarks indicate that the items are boolean properties and the 1's indicate that they are integer properties.

How can the Application be Improved?

Although all the controls do their jobs, they try to do their jobs when it is inappropriate to do so. For example,

- If the TextField is blank, clicking the Add button adds a blank row to the ListBox and clicking the Connect button tries to open a non-existent URL.
- If no item is selected in the ListBox, the Delete button tries to delete a non-existent row.

We should make it impossible for the buttons to be clicked unless the conditions are right or, at least, give the user some appropriate feedback.

1. Choose File ▶ Exit on Windows or Linux (or MyApplication.debug ▶ Quit on Mac OS X) to return to the REAL Studio IDE and select the Window editor tab.
2. In the Code Editor browser area, open the Connect button's Action event.
   It currently reads:

```
ShowURL SelectedURL.Text
```

   We need to make this conditional — so that the line is executed only when there is some text in the TextField. For now, we'll have to assume that the user knows how to enter a valid URL or email address.
3. Modify the code to read:

```
If SelectedURL.Text <> "" then
  ShowURL SelectedURL.Text
Else
  MsgBox "Please enter a URL or email address."
End if
```

   There is no space between the double quote marks in the first line or between the less than and greater than signs.

   This code first tests to see whether the contents of the TextField is blank. The "<>" symbol means "not equal to" and the two quotes with nothing in between specify blank text. This If statement tests whether the TextField is not empty. If the TextField has text in it, the ShowURL method is called normally.

   The Else statement handles the case when the TextField is empty. If the TextField is blank, we use the MsgBox command to display an alert box with an informative message.
4. Expand the DeleteURL item in the Code Editor and select the Action event.

We need to follow the same logic. This code should run only if the user has selected an item in the ListBox—not all the time.

5 Modify the code to read as follows:

```
If ListURL.Text <> "" then
  ListURL.RemoveRow ListURL.ListIndex
Else
  MsgBox "Please select an item in the list."
End if
```

6 Expand the AddURL item in the Code Editor and select the Action event.

In this case, we can add an extra test. If the item is already selected, it doesn't need to be added, so we can test for that condition as well. The first test prevents the Add button from adding a blank row and the second test prevents it from adding a duplicate row.

7 Modify the code to read as follows:

```
If SelectedURL.Text <> ListURL.Text then
    ListURL.Addrow SelectedURL.Text
Else
  MsgBox "Please enter a different URL or email address."
End if
```

Next, we want to prevent the user from using the Add and Connect buttons if there is nothing in the TextField.

8 Expand the SelectedURL item in the Code Editor and select the TextChange event handler in the Browser area of the Code Editor.

The TextChange event runs whenever the text in the TextField has changed. We want to take action based on the TextField's state just after the text has changed. If the TextField is now blank, we will disable these two buttons so the user can't use them. And, if there is text in the TextField, we will enable the buttons and make the Connect button the default button. The default button has an outline around it on Windows and Linux and it "throbs" on Mac OS X.

9 Enter the following code into the TextChange event.

```
If Me.Text <> "" then
  ConnectURL.Enabled=True
  ConnectURL.Default=True
  AddURL.Enabled=True
Else
  ConnectURL.Enabled=False
  ConnectURL.Default=False
  AddURL.Enabled=False
End if
```

One thing you notice about this code is that the first line uses the pronoun "Me" instead of the name of the control. "Me" always refers to the control the code belongs to; since we are inside the TextField, SelectedURL, "Me" refers to SelectedURL. In other words, this line is the same as if we wrote `If SelectedURL.Text…`.

The last step is to modify the code for the ListBox. This code needs to test whether the user has highlighted an item before trying to copy text into the TextField. It also should manage the Delete and Connect buttons. There's no reason the user should be able to click Delete if no item is selected.

10 Expand the ListURL item in the Code Editor and select the Change event.

11 Modify the code to read as follows:

```
If Me.Text <> "" then
  SelectedURL.Text=ListURL.Text
  ConnectURL.Enabled=True
  DeleteURL.Enabled=True
Else
  DeleteURL.Enabled=False
End if
```

12 Choose Project ▶ Analyze Project to see if you made any syntax errors.
Fix any errors and, when all is well, proceed to test the application.

13 Click the Run button  in the Main Toolbar to test it out.
When you first open the application, you can test the alert messages that you've put in each PushButton. Then add a URL and see how the application behaves.

14 When you're finished, quit out of the test application and return to the REAL Studio IDE.

If the Application Doesn't Run

This section introduces the If…Else…End if statement. The Code Editor automatically indents the code within an If statement to make it easy to check your code. Be sure that your code is indented as shown in the instructions. Be sure there is an "End If" statement for each "If" and you didn't type "Endif" instead of "End if" or "end if". Capitalization doesn't matter, but the space between "End" and "if" does. Also, check that each line that begins with an "If" statement ends with "then". Also, you should not leave a blank space between the double quote marks or between the less than and greater than signs.

Building A Standalone Application

Now that you have a finished version of the application, you're ready to create a standalone application. The *standalone* version of the application runs just like the application you've been testing, but it doesn't require the REAL Studio application itself. It can be double-clicked from the desktop, just like a commercial application.

The Professional version of REAL Studio can create standalone applications for the Windows, Macintosh, and Linux platforms. The Personal version of REAL Studio

builds standalone applications only for the platform on which the REAL Studio IDE is running. It also allows you to build demo versions for the platforms on which REAL Studio is *not* running. A demo version is fully functional but quits automatically after five minutes.

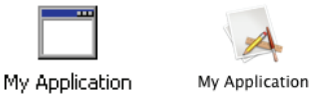
Building Your Application

Building a stand-alone version of your project as an application couldn't be easier than it is in REAL Studio.

1 Click the Build button  in the Main Toolbar.

When you create a standalone application, REAL Studio creates a new folder called "Builds" and creates subfolders for every platform for which you create a build. It puts the standalone application in the proper subfolder. It will open it up and highlight the application.

Figure 29. The URL Manager built application icon (Windows and Macintosh).



By default, it uses the name "My Application" and uses your platform's generic application icon. Both the name and icon can be customized easily.

2 Double-click the "My Application" icon and try out the program. When finished, choose Quit to put away the URL Manager and go back to the REAL Studio project.

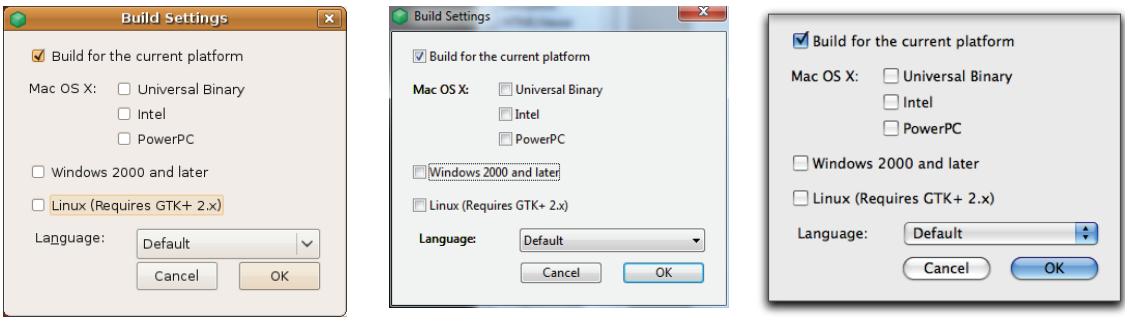
Customizing the Standalone Application

Before building a standalone application, you can set several options. For example, you can change the application's name and build for other platforms.

You can choose the platforms on which you build your application in the Build Settings dialog box.

1 Choose Project ▶ Build Settings.
The Build Settings dialog box appears, with your platform selected.

Figure 30. The Build Settings dialogs (Linux, Vista, Mac OS X).



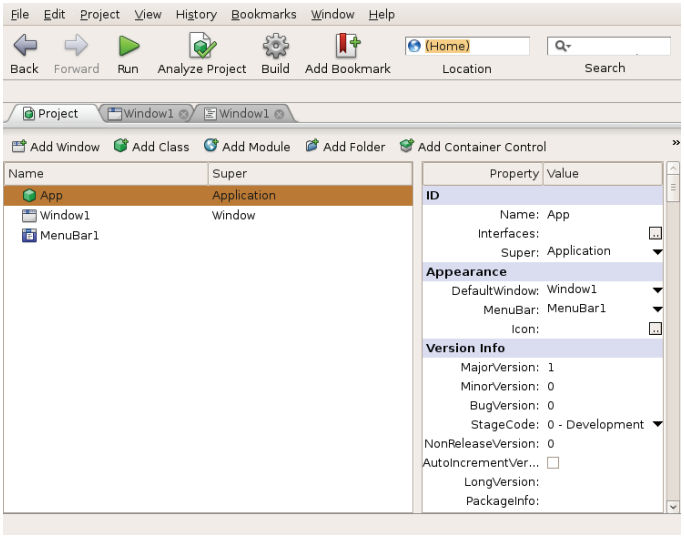
The Build Settings dialog selects the platform that is currently running.

2 If you have other computer platforms available, select them and click OK.

You can also set the properties of the standalone application, including its name. You do this by setting some properties of the *App* object. You access this object from the *Project Editor*.

1 Click the Project tab in the Tabs bar.
The Project Editor appears.

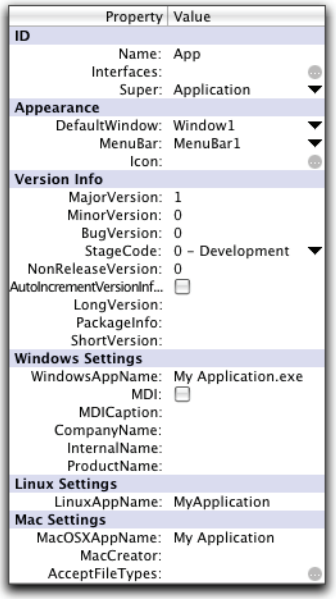
Figure 31. The Project Editor for the URL Manager application.



The Project Editor contains a list of all of the major items that make up your REAL Studio application. For example, it includes items for all the windows that your application uses, the menu bars, and objects such as sounds, pictures, databases, and movies. By default, it includes an item for the application's main window, *Window1*, its default menu bar, *MenuBar1*, and an item associated with the application as a whole, *App*. You double-click an item in the Project Editor to edit or view it.

When you highlight the App object, as is shown in Figure 31, The Properties pane shows the App object's properties.

Figure 32. The properties of the App class.



In the Windows, Linux, and Mac settings areas, you set the name of the built application and, in most cases, some other properties.

2. Enter **URL Manager.exe** for Windows or **URL Manager** for Macintosh and Linux as the name of the application for the OS that you're using.
3. Click the Build button  in the Main Toolbar.
   REAL Studio compiles the application, saves the built application to a file on your computer, and opens the window containing the built application.
   REAL Studio creates a folder named "Builds" that contains all of the built applications. Inside the Builds folder, there are separate subfolders for each platform that you selected in the Project Settings dialog box.

   Quit out of the REAL Studio IDE (File ▶ Exit on Windows and Linux or REAL Studio ▶ Quit on Mac OS X).
4. Double-click the new application and try it out.
5. If you built versions for other platforms, drag them to those platforms as well and try them out.

What's Next

The *Tutorial* shows how easy it is to develop a simple application. But there is much more to REAL Studio. As hinted in the Project Editor, REAL Studio includes a Menu Editor and can support multiple windows of different types.

REAL Studio is a fully object-oriented development system. With it you can create classes (reusable objects) that encapsulate custom functionality. You can add them to the project so that they can be reused in many places in the project and save them

to disk. In the *User's Guide*, you will learn about the object-oriented development system in detail. The *Language Reference* is a comprehensive listing of all the programming elements in the REAL Studio language.

Also, be sure to check out the example projects that ship with REAL Studio and the REAL Studio web site at http://www.realsoftware.com for other tutorials and how-to's.