XOJO

Web Tutorial

Xojo, Inc.

Chapter 1

Introduction

This tutorial show you how to create a web application.

Section 1

About the Web Tutorial

STARTING XOJO

1. **Download the installer for your operating system from: http://www.xojo.com/download**
2. **Run the installer.**
3. **Run Xojo.**
4. **In the Project Chooser, select Web and click OK.**

This Web Tutorial is for people who are new to programming and new to Xojo. It is an introduction to the Xojo development environment and will lead you through the development of a real web application.

It should take you about an hour to complete this tutorial.

Note: If you have experience with other programming languages, you'll also want to check out the User Guide and Language Reference.

information. Use of a term in this book should not be regarded as affecting the validity of any trademark, registered trademark, or service mark. Xojo, Inc. is not associated with any product or vendor mentioned in this book.

Section 2

Presentation Conventions

The Tutorial uses screen snapshots taken from the Windows, OS X and Linux versions of Xojo. 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 Windows, OS X, and Linux graphical user interfaces.

- **Bold 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 *Xojo User Guide*, are italicized.

- When you are instructed to choose an item from one of the menus, you will see something like “choose File ↠ New Project”. This is equivalent to “choose New Project from the File menu.”

- Keyboard shortcuts 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 OS X, 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 OS X and the press the “O” key. You release the modifier key only after you press the shortcut key.

- Something that you are supposed to type is quoted, such as “GoButton”.

- Some steps ask you to enter lines of code into the Code Editor. They appear 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.

- Of course, you can copy and paste the code as well.

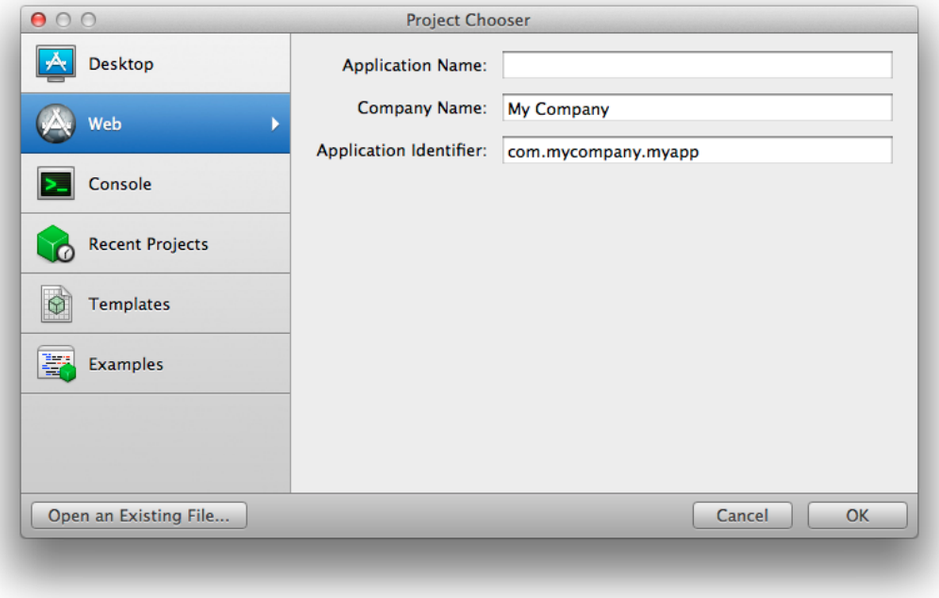
Whenever you run your application, Xojo first checks your code for spelling and syntax errors. If this checking turns up an error, an error pane appears at the bottom of the main window for you to review.

Getting Started

If you haven't done so already, now is the time to start Xojo.

Double-click the Xojo application icon to start Xojo. After it finishes loading, the Project Chooser window appears.

Figure 1.1 Project Chooser Window



Xojo lets you build three different types of applications (Desktop, Web and Console). For this Tutorial, you are building a web application, so click on Web.

You should now see three fields to specify: Application Name, Company Name and Application Identifier.

Application Name is the name of your application. This will be the filename of the actual application file that gets created.

Company Name is the name of your company. You may choose to leave this blank.

Application Identifier is a unique identifier for this application. It will automatically populate using what you enter for the Application and Company Names, but you can also change it to whatever you want.

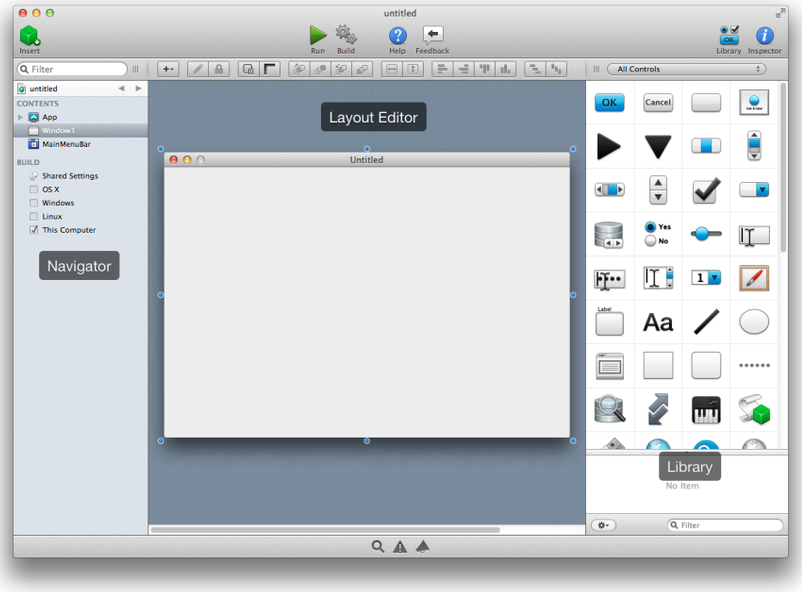
Enter "TaskManager" as the Application Name. You can leave Company Name as it is or change it.

Click **OK** to open the main Xojo window (called the Workspace), where you will begin designing your application.

Workspace

Xojo opens the Workspace with the default window for your application selected in the **Navigator** and displayed in the **Layout Editor**.

Figure 1.2 The Xojo Main Window



Navigator: The area on the top left shows you all the items in your project. By default you can see WebPage1 (which is selected), the App object and the Session object. You use the Navigator to navigate within your project.

Layout Editor: The center area is the Layout Editor. You use the Layout Editor to design the user interface for the windows in your application. It shows the window and previews how it looks when the application runs. In this illustration, the window is blank because you haven't yet added any user interface controls from the **Library**.

Library: The area on the right is the Library and shows the controls and interface elements that you can add to a window or to the project. You design the window by dragging controls from the Library to the window. You can also add a control to the window by double-clicking it.
You can change how the controls display in the Library by clicking the small gear icon  and choosing a different setting.

Note: If the Library is not visible, click the Library button on the toolbar to show it.

Inspector: Not shown in the above illustration is the Inspector, which allows you to see and change the properties for the

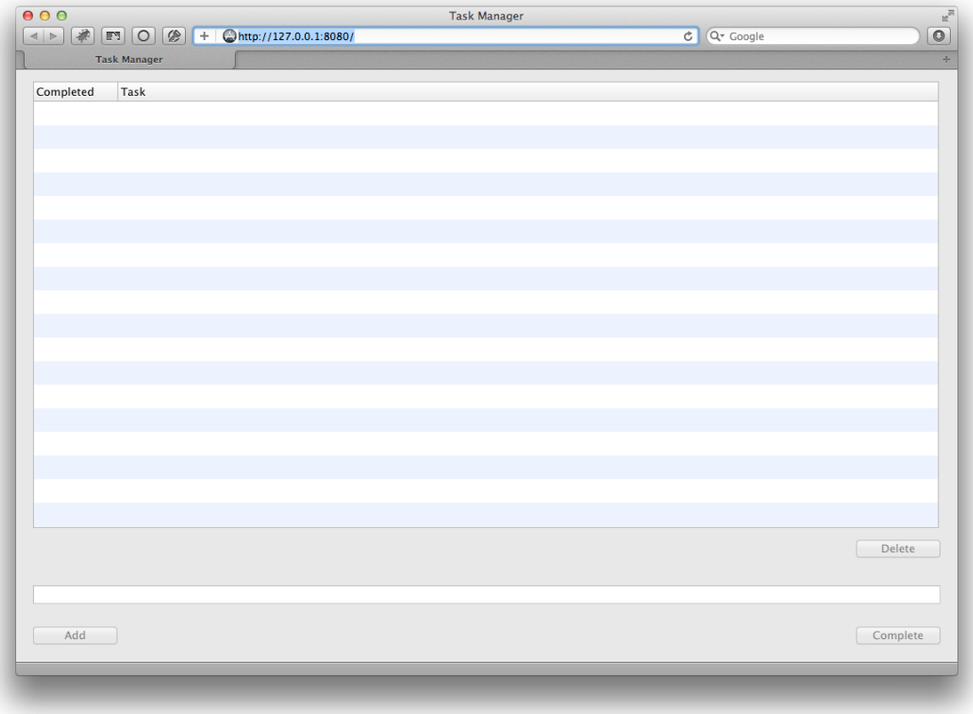
selected control. This area of the Main window is shared with the Library. You can show the Inspector by clicking the Inspector button on the toolbar. The Inspector shows information about the selected item in the Navigator or Editor. The contents of the Inspector changes as you click on different items. You change an Inspector value by entering a new value in the field to the right of the field label.

About the Application

Task Manager

In this tutorial you will create an application to track tasks as shown.

Figure 1.3 Task Manager Web Application



You enter tasks in the text field and click Add to add them to the list. You can click on individual tasks in the list to delete them or mark them as complete.

Task Manager uses three controls:

WebTextField: A **WebTextField** control is used to enter text. In this project, the task to add is entered into a WebTextField at the bottom of the window.

WebButton: A **WebButton** is used to trigger an action. This project uses several buttons to perform different actions.

WebListBox: A **WebListBox** is used to display a list of data. In this project, it is what displays the tasks entered in the TextField.

The next sections walk you through creating the user interface and adding the necessary code to make the application work.

Chapter 2

Designing the User Interface

Now you will design the user interface for the Task Manager web application.

Task List

Adding the Task List

You should have Xojo running and WebPage1 open in the Layout Editor. If not, please refer to Chapter 1, Sections 3 and 4.

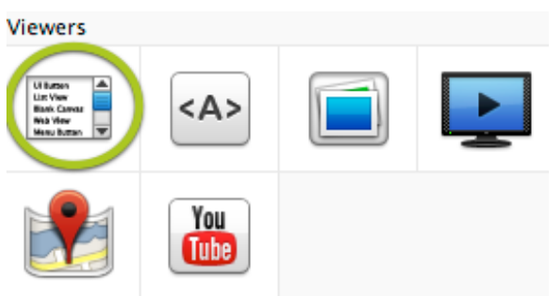
You are now going to add a Listbox to the web page. The Listbox is used for storing the tasks.

1. Add the Listbox:

   In the Control Library, click on the Listbox and drag it to the top-left corner of the Layout Editor.

   As you get close to the edges of the web page, you will see alignment indicators that help you position the control. Drop the Listbox when you are happy with its position on the web page.

Figure 2.1 ListBox Control

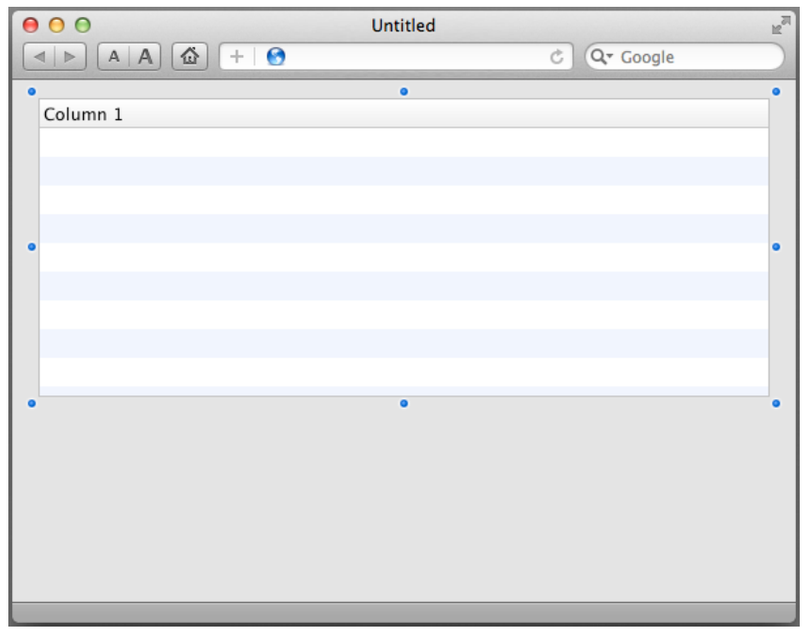


2. Resize the Listbox:

   Click on the Listbox so that the resizing handles appear. Grab the handle in the bottom-right corner and drag it to enlarge the Listbox to fill the top 2/3 of the web page.

3. Your web page should now look like this:

Figure 2.2 Web Page Layout with List Box



Buttons

Adding the Buttons

Now you will add the three buttons needed by Task Manager to the web page.

The Delete button removes tasks from the Listbox, the Add button adds tasks to the Listbox and the Complete button marks tasks in the Listbox as completed.

1. Add the Delete Button:

   In the Library, click on the Button control and drag it to the web page below the lower-right corner of the Listbox.

   Figure 2.3 Button Control

   Use the alignment indicators to help you position the button so that it lines up with the right edge of the Listbox

2. Add the Add Button:

   In the Library, click on the Button control and drag it to the web page near the bottom-left corner.

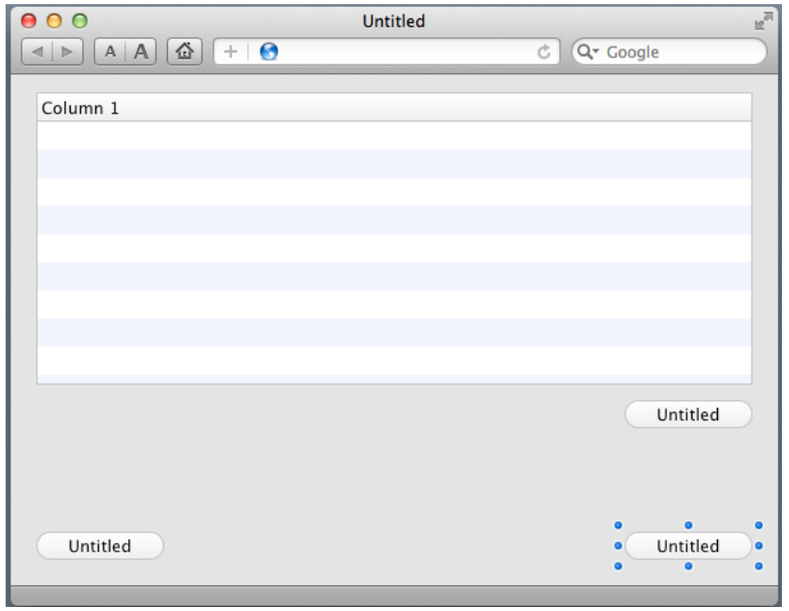
   Again, take advantage of the alignment indicators to help you position the button.

3. Add the Complete Button:

   In the Library, click on the Button control and drag it to the web page near the bottom-right corner.

4. Your web page now looks like this:

   Figure 2.4 Web Page Layout with the Buttons Positioned

Text Field

Add the Text Field

The Text Field is where the user types the Task to add to the list.

1. In the Library, click on TextField and drag it to the web page so that it is between the delete and add buttons.

2. Resize the Task field. Select the rightmost drag handle and drag the TextField so that it is the same width as the Listbox.

   Use the alignment indicators as guides to help you line everything up correctly.

3. Your web page now looks like this:

Figure 2.5 Text Field Control

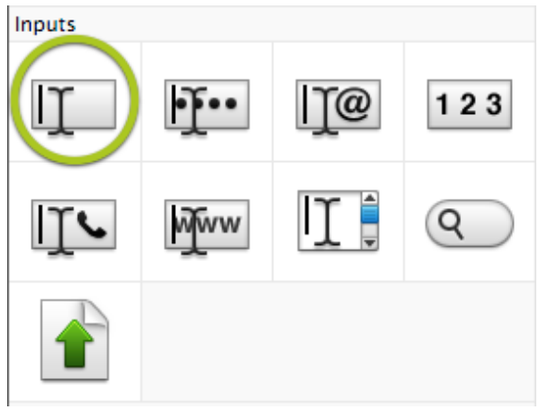
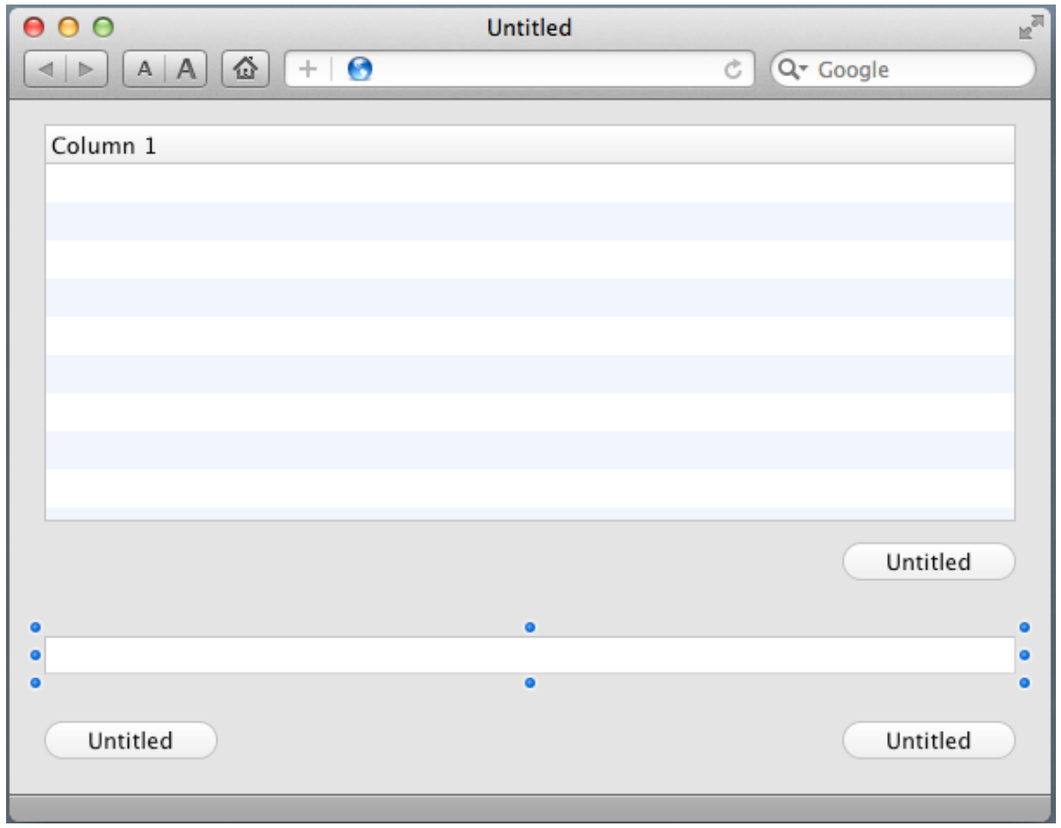


Figure 2.6 Completed Web Page Layout



Section 4

Properties

What is a Property?

A **property** is a value of a class. Changing property values allows you to change the behavior of the class.

For this project, you want to change various properties for the web page and its controls. Some of the things you need to do are:

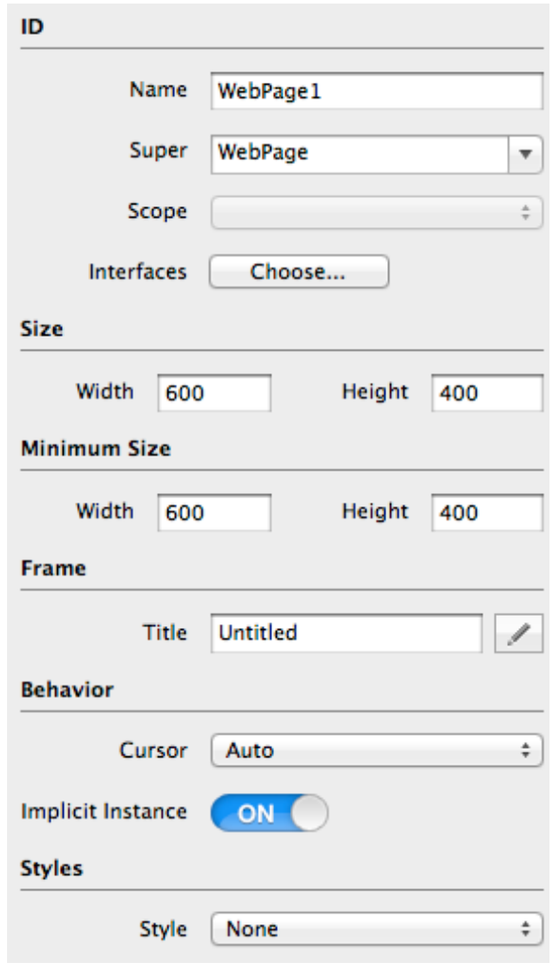
- Rename all controls (and the web page) so that they describe their behavior and are easy to refer to in code.
- Add a Caption to the Buttons.
- Set Locking properties so that the controls resize properly when the web page size is changed.

Inspector

The **Inspector** is used to change the window and control properties. It shares the same area on the right of the main window as the Library. In order to show the Inspector, click the Inspector button on the toolbar.

You can toggle between the Library and Inspector using ⌘-L and ⌘-I and on OS X or Ctrl-L and Ctrl-I on Windows and Linux.

Figure 2.7 Web Page Properties in the Inspector



Web Page Properties

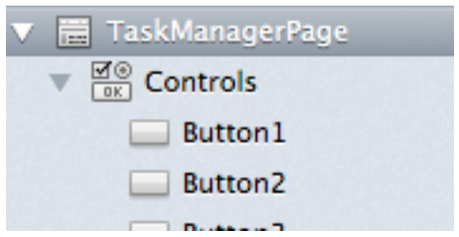
If you haven't already, display the Inspector by clicking the Inspector button on the toolbar.

You need to change the **Name** and **Title** properties of the web page:

1. First, in the Layout Editor, click on the title bar of the web page to select it. The Inspector pane now shows the properties for the web page.

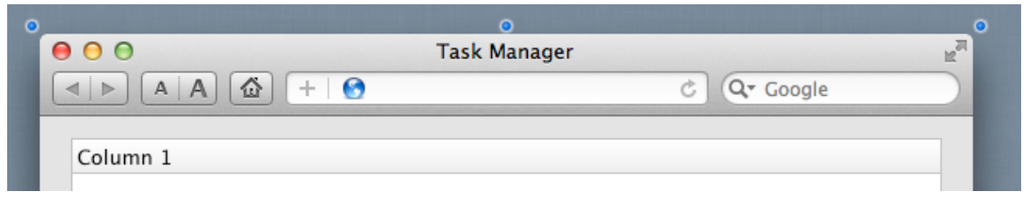
2. In the Name field (located in the ID group), change the name from "WebPage1" to "TaskManagerPage". Press *Return* to see the name change in the Navigator.

Figure 2.8 Name Change in Navigator



3. In the Title field (located in the Frame group), change the name from "Untitled" to "Task Manager". Press *Return* to see the name change in the title bar of the web page.

Figure 2.9 Web Page Layout with Updated Title Bar



Section 6

ListBox Properties

The Listbox is where the tasks that your user enters are displayed and stored. You need to change the following properties: **Name, ColumnCount, Default Value, ColumnWidths** and **Locking.**

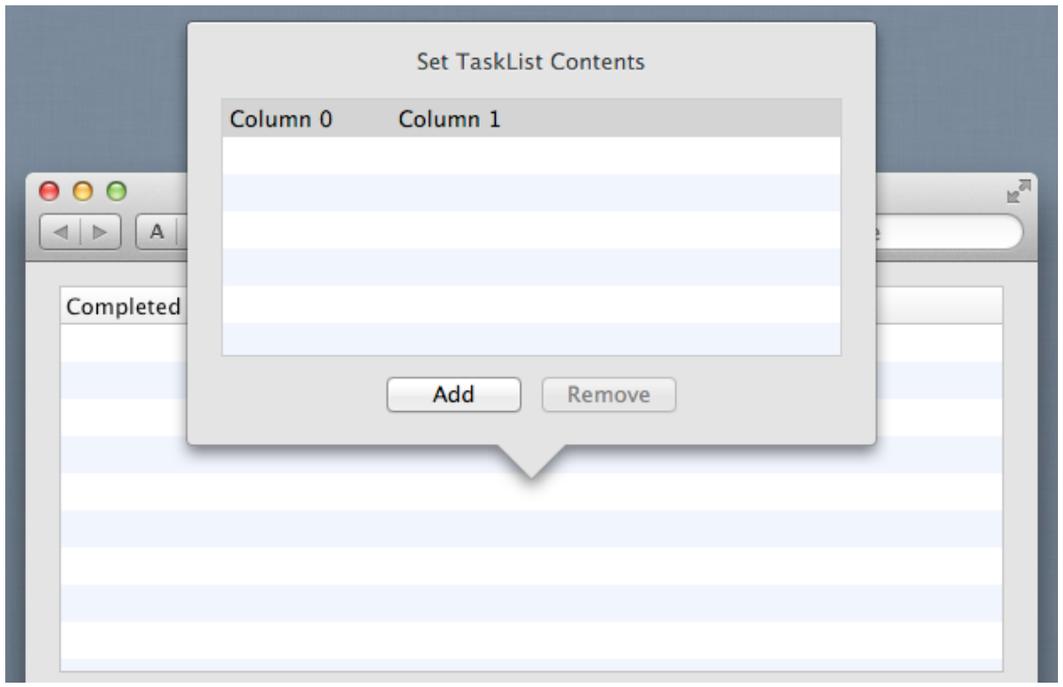
1. First, in the Layout Editor, click on the Listbox to select it. The Inspector pane now shows the properties for Listbox.

2. In the Name field (located in the ID group), change the name from “Listbox1” to “TaskList”. Press *Return* to see the name change in the Navigator.

3. The Listbox has two columns, one to show the completed status and another to show the name of the task. In the ColumnCount field, change the value from “1” to “2”. Press Return to see the Listbox appear with two columns on the web page layout.

4. You want to change the column headers to describe the data in the list. Click the “Set Default Value of TaskList...” button on the Layout Editor toolbar (). This opens the Value Editor popout window:

Figure 2.10 Default Value for TaskList



a. Click on “Column 0” in the header to select it and then click on it again to edit its value. Type “Completed” and press *Return*.

b. Click on “Column 1” in the header to edit its value. Type “Task” and press Return.

c. Click on the “Set Default Value of TaskList...” button again to close the popout window. “Completed” and “Task” now appears as the headers for TaskList.

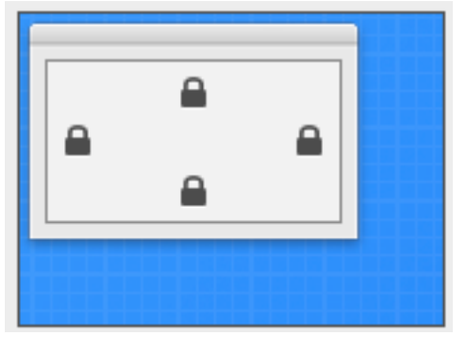
5. Since Completed is only going to contain a simple checkmark when the task is marked as completed, it can be narrower. In the ColumnWidths field, change the value from “*” to “100,*”. Press *Return* to see the column widths change on the web page.

   Using “100,*” tells the Listbox that the first column should always be 100 pixels wide and that the rest of the columns share the available width.

6. Lastly you need to make changes to the locking so that the Listbox gets larger or smaller as the web page size changes.

   In the Locking group look at the image that shows the web page with small locked icons for the top and left and small unlocked icons for bottom and right. Click the locks so that top, left, bottom and right are all locked.

Figure 2.11 Locking for TaskList



Button Properties

The three buttons are used to perform actions. You need to change the following properties for each button: **Name, Caption** and **Locking.**

Delete Button

The Delete button is used to remove tasks from the TaskList.

1. First, in the Layout Editor, click on the Delete button to select it (this is the button directly below the Listbox). The Inspector now shows the properties for WebButton.

2. In the Name field (located in the ID group), change the name from “Button1” to “DeleteButton”. Press *Return* to see the name change in the Navigator.

3. In the Caption field (located in the Appearance group), change the name from “Untitled” to “Delete”. Press *Return* to see the name change on the button in the web page.

4. Now you need to make changes to the locking so that the Delete button stays on the right side of the web page when the web page resizes.

In the Locking group look at the image that shows the web page with small locked icons for the top and left and small unlocked icons for bottom and right. Click the locks so that right and bottom are locked and left and top are unlocked.

Figure 2.12 Locking for DeleteButton

Add Button

5. The Add button is used to add the task entered in the TextField to the Task List.

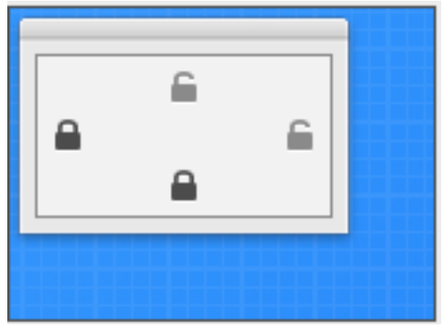
1. In the Layout Editor, click on the Add button to select it (this is the button on the far left of the web page below the TextField). The Inspector now shows the properties for PushButton.

2. In the Name field (located in the ID group), change the name from “Button2” to “AddButton”. Press *Return* to see the name change in the Navigator.

3. In the Caption field (located in the Appearance group), change the name from “Untitled” to “Add”. Press *Return* to see the name change on the button in the web page.

4. Now you need to check the locking so that the Add button stays on the bottom of the web page when the web page resizes.

   Figure 2.13 Locking for AddButton

   In the Locking group look at the image that shows the web page with small locked icons for the top and left and small unlocked icons for bottom and right.
   Click the locks so that left and bottom are locked and top and right are unlocked.

Complete Button

The Complete button is used to mark a task as completed.

In the Layout Editor, click on the Complete button to select it (this is the button directly below the TextField on the right). The Inspector now shows the properties for WebButton.

In the Name field (located in the ID group), change the name from “Button3” to “CompleteButton”. Press *Return* to see the name change in the Navigator.

In the Caption field (located in the Appearance group), change the name from “Untitled” to “Complete”. Press *Return* to see the name change on the button in the web page.

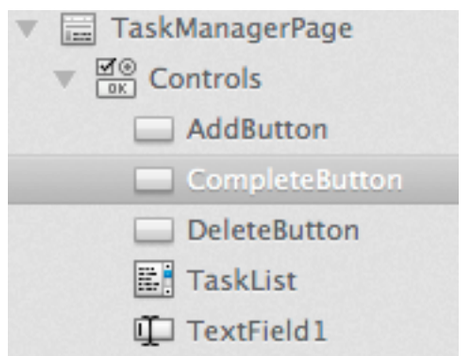
Now you need to make changes to the locking so that the Complete button stays on the right side of the web page when the web page resizes.

In the Locking group look at the image that shows the web page with small locked icons for the top and left and small unlocked icons for bottom and right.
Click the locks so that right and bottom are locked and left and top are unlocked.

Figure 2.15 Locking for CompleteButton

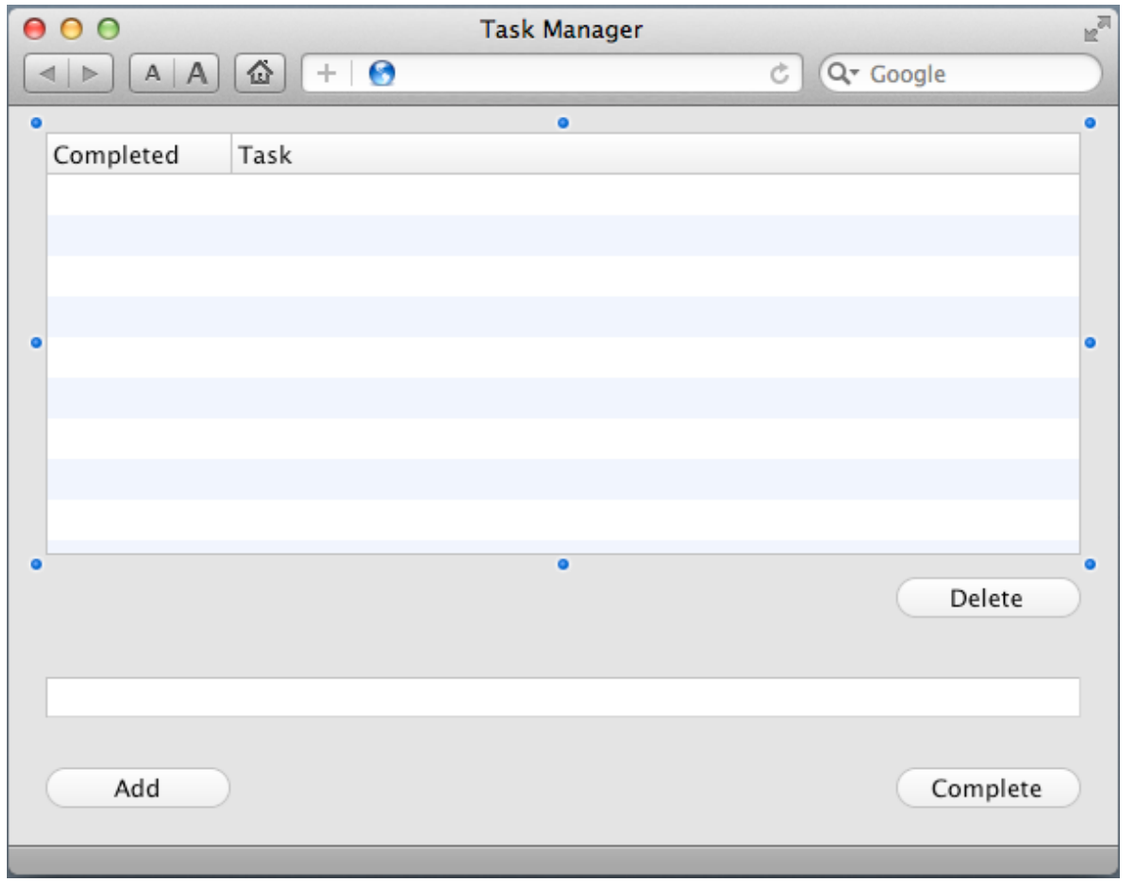
In the Project List, the newly renamed controls show under the Controls for TaskManagerPage.

Figure 2.14 Controls Appearing in Navigator



The web page layout now looks like this:

Figure 2.16 Web Page Layout with Button Captions



Text Field Properties

The TextField is where your user will type the task to add to the list. You need to change the following properties: **Name** and **Locking.**

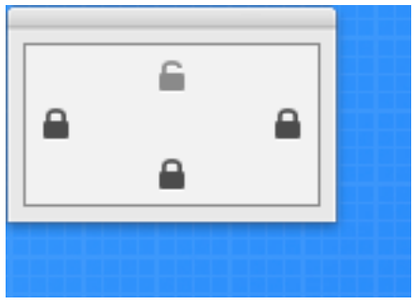
1. In the Layout Editor, click on the WebTextField to select it. The Inspector now shows the properties for WebTextField.

2. In the Name field (located in the ID group), change the name from “TextField1” to “TaskField”. Press *Return* to see the name change in the Navigator.

3. Now you need to make changes to the locking so that the TextField gets larger or smaller when the web page resizes.

   In the Locking group look at the image that shows the web page with small locked icons for the top and left and small unlocked icons for bottom and right.

Figure 2.17 Locking for TaskField



Click the locks so that left, bottom and right are locked and top is unlocked.

Testing the Project

Saving Your Project

You should save your work periodically and always before running your project.

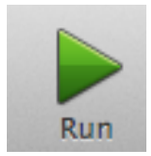
1. Save the project by choosing File ➙ Save.

2. Name the project “TutorialWeb” and click Save.

Running Your Project

Now you can test your finished application:

Your user interface layout is now complete, so it’s time to try it out. Click the Run button in the toolbar to run the project.

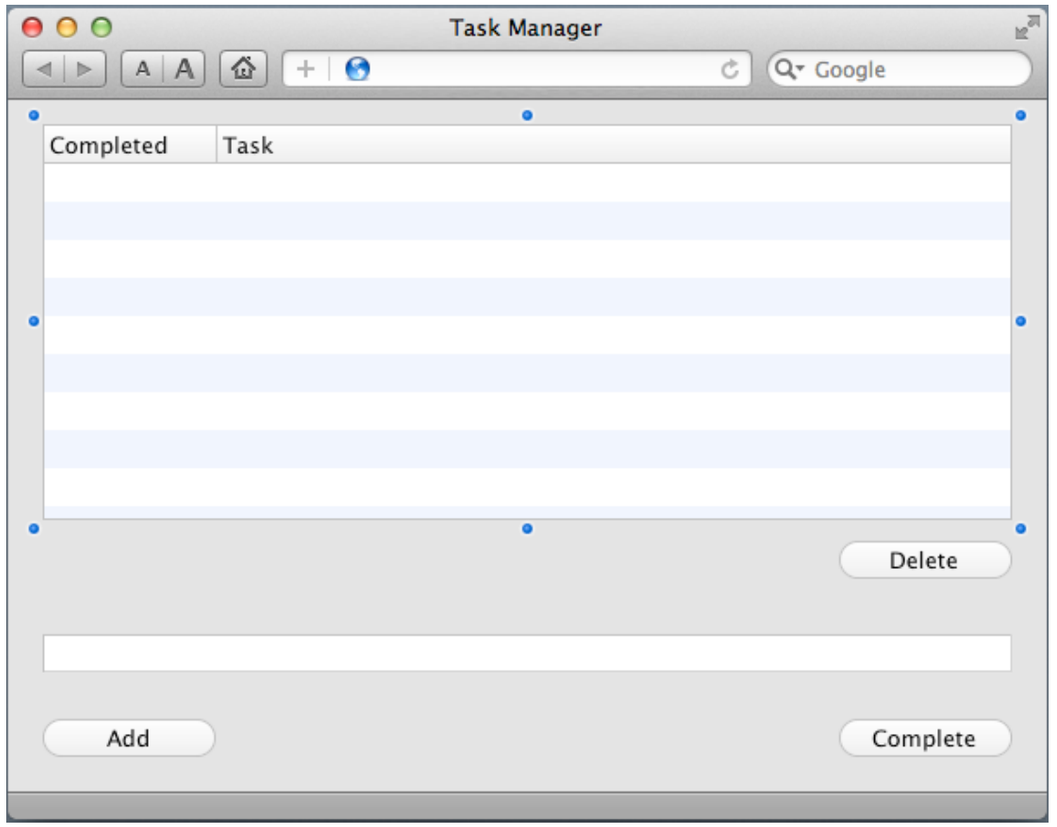


When Task Manager is displayed, you can interact with the buttons by clicking on them, you can type in the TextField and you can resize the web page to see the buttons reposition themselves.

Your application doesn’t yet do anything. For that you need to add code, which is the subject of the next chapter.

Close the browser tab or window to return to the Layout Editor.

Figure 2.18 Completed Task Manager Web Page Layout



Chapter 3

Adding Code

The final step in creating your application is to add the code.

Add Button

Adding Code to the Add Button

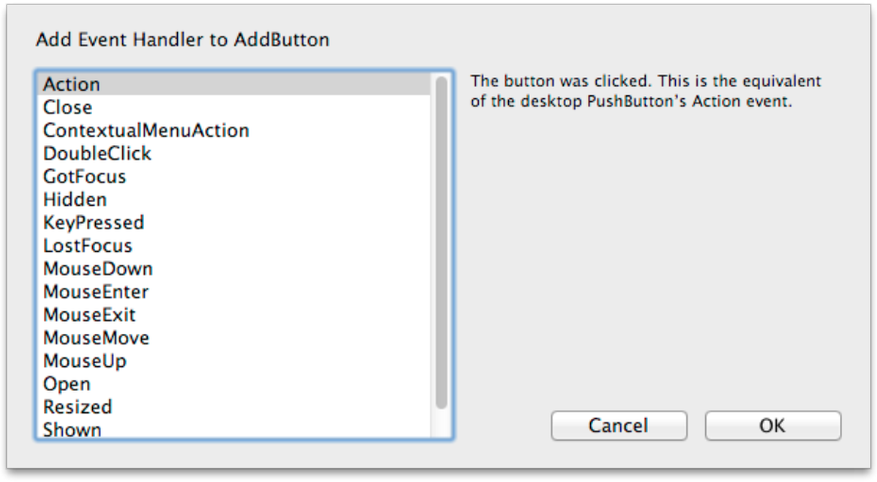
The Add button adds tasks to the list. The code you add to the button needs to take what was typed in TaskField and add it as a new row to the list.

Follow these steps to add the code:

1. On the web page, double-click the **AddButton** control, labelled “Add”.

The **Add Event Handler** window appears. When a user clicks on a WebButton, the **Action** event handler is called. This means you want to add your code to the Action event handler, so select Action from the Event Handler list and click OK.

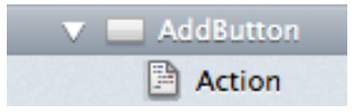
Figure 3.1 Add Event Handler Dialog



This displays the Code Editor.
Also notice the Navigator updates to show the Action event underneath the AddButton control.

2. Now you need to get the task that was typed into the Task field. You might think you could get the task just by referring to the name of the field, TaskField. That is close, but not quite what you want.

Figure 3.2 Action Event Handler



What you instead need is a property of TaskField. When you need to refer to a property of an object, you use 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*. This is something called “dot” notation and is commonly used in object-oriented programming.

In this case the object is **TaskField** and the property you want is **Text** (use the Language Reference to find out about all the properties available to TextFields).

The syntax looks like this:

```
TaskField.Text
```

3. To actually add a row to a Listbox, you use the AddRow method. You already know how to get the text in the Task field from step 2. Combine the two to get this code:

   ```
   TaskList.AddRow("", TaskField.Text)
   ```

   As you have seen before, objects can have properties. And as you now see with TaskList, objects can also have methods. AddRow is one of many methods available to Listboxes.

   The above command adds values to the two columns in TaskList. The first column contains the completed status, so it is initially set to blank. The second column contains the name of the task.

4. Save the project by choosing File ↠ Save.

5. Run the app to test it out. Type tasks in the task field and click the Add button to see them appear in the task list.
   Close the browser tab or window to return to the Code Editor.

Complete Button

Adding Code to the Complete Button

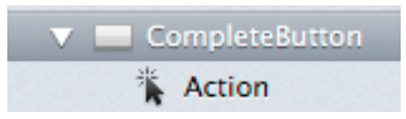
When the user clicks the Complete button, the selected task in the Listbox should be marked as completed. This is indicated by showing a checkmark (✓) in the Completed column.

Follow these steps to add the code:

1. On the web page, double-click the **CompleteButton** control, labelled “Complete”.

   The **Add Event Handler** window appears. When a user clicks on a WebButton, its **Action** event handler is called. This means you want to add your code to the Action event handler, so select Action from the Event Handler list and click OK.

   Figure 3.3 Action Event Handler

   Notice the Navigator updates to show the Action event underneath the CompleteButton control and the code editor displays.

2. To change a row, you first need to know what row is selected. In a Listbox, the currently selected row is contained in the **ListIndex** property.

3. To set the value in a particular cell of a Listbox, you use the Cell property, specifying the row and column. The code looks like this:

   ```
   TaskList.Cell(TaskList.ListIndex, 0) = "✓"
   ```

   This code put the checkmark character in column 0 (the completed column) of the currently selected row.

4. Run the app and add a few sample tasks. Now click on a task and click the Complete button. A checkmark appears in the Completed column.

   Close the browser tab or window to return to the Code Editor.

Section 3

Delete Button

Adding Code to the Delete Button

The Delete button is used to remove tasks from the list. The code you add to the button needs to determine the selected row in the list and remove it from the list.

Follow these steps to add the code:

1. On the web page, double-click the **DeleteButton** control, labelled “Delete”.

   The **Add Event Handler** window appears. As you learned with the other buttons, you want to use the **Action** event handler to have your code run when the user clicks on a button. Select Action from the Event Handler list and click OK.

   Notice the Navigator updates to show the Action event underneath the DeleteButton control and the code editor displays.

2. Since the selected row will be deleted, you again want to use the **ListIndex** property.

3. Use the Listbox method **RemoveRow** to remove a row from the Listbox. You pass RemoveRow the row number to remove as a parameter. So your code looks like this:

   ```
   TaskList.RemoveRow(TaskList.ListIndex)
   ```

4. Save the project by choosing File ➙ Save.

5. Run the app and add a few sample tasks. Now click on a task in the Task List and click the Delete button. The task is removed from the list.

Debugging

Finding Bugs

Although your app works just fine, there are a couple of lingering bugs that need addressing. A bug is when your code or application does something unexpected, often leading to a crash. Have you figured out what the problem is?

Here's a hint: What happens if you you click on the Complete or Delete buttons but have not selected a task? Try it.

1. Run the app and click on the Complete button without doing anything else.

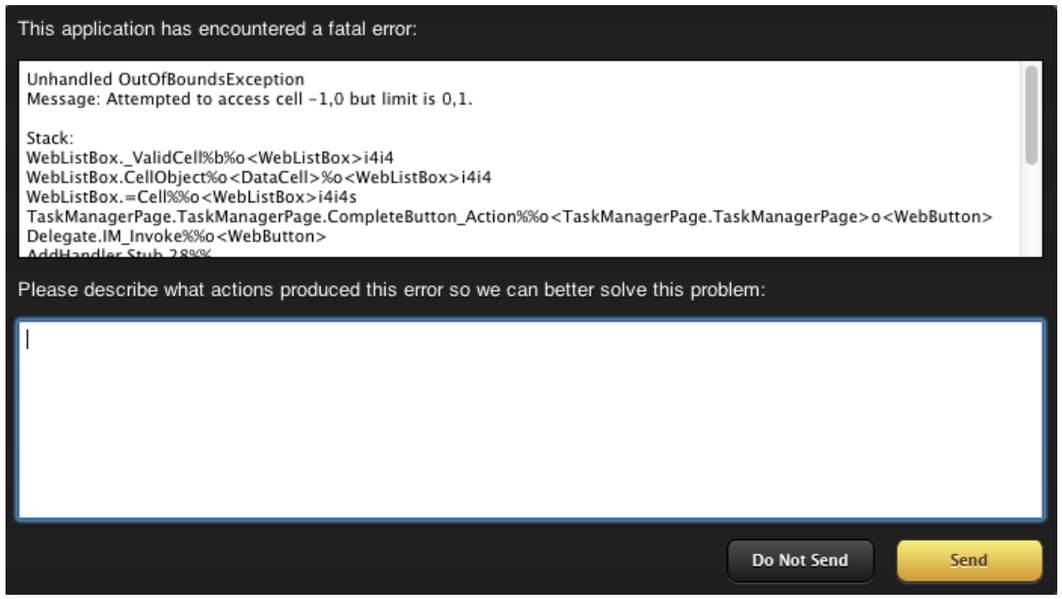
Figure 3.4 Debugger Stopped at a Line of Code that Caused an Error

```
Sub Action()
TaskList.Cell(TaskList.ListIndex, 0) = "✓"
End Sub
```

Your app will switch to the Debugger with a line of code highlighted. Your code crashed with an OutOfBoundsException and you are now in the debugger.

The error occurred because you attempted to remove (or complete) a row that does not exist. When no row is selected in the Listbox, the ListIndex property returns -1. Since this is not a valid row in the Listbox, the Cell command raises an OutOfBoundsException.

Figure 3.5 A Run-Time Error Displayed in the Browser



2. Click the Resume button in the debugger toolbar , to see the actual error message.

3. Close the browser tab or window to return to the Editor.

Nobody wants buggy code. Luckily it is easy to prevent this bug from occurring. Essentially, you want to make sure a row is selected before you attempt to Delete or Complete a task.

1. The code to do this uses what you have already learned.

   This is the code for Action event handler of the DeleteButton:

   ```
   If TaskList.ListIndex >= 0 Then
     TaskList.RemoveRow(TaskList.ListIndex)
   End If
   ```

2. The code for the Complete button is similar:

   ```
   If TaskList.ListIndex >= 0 Then
     TaskList.Cell(TaskList.ListIndex, 0) = "✓"
   End If
   ```

3. In both cases, the code verifies that a row is selected by checking the ListIndex property to ensure that it contains a valid row before the actual method is called.

4. Save the project by choosing File ⇥ Save.

5. Run the project again and click the Complete button without selecting a row in the task list. No more crash!

Chapter 4

Next Steps

Did you think you were done? Not quite yet.

Section 1

Testing Task Manager

You Still Have to Test

Just because you have finished coding your application, doesn't mean you are finished. A good developer always thoroughly tests their applications to look for possible problems.

You already found and fixed one problem (clicking Delete or Complete when no row was selected). Do you think there are other problems to fix?

Run your application and play around with it a bit. Make a note of things you want to change. In the next section, you will make some improvements to Task Manager.

Improvements

Button Usage

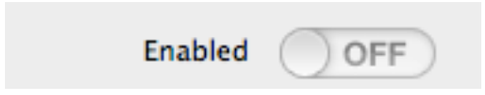
Did you notice that there are times when the buttons in Task Manager probably should not do their action? For example, the Complete button should not try to mark a task as completed if one is not selected. Right now you can click it, but nothing happens. Also, you are not going to want to add a task to the list if nothing has been entered in the task field.

There are several ways to accomplish this, but one way is to disable the buttons when they should not be used.

Follow these steps to add this improvement:

1. On the web page, select **CompleteButton**, labeled "Complete". In the Inspector, turn the Enabled property (in the Appearance group) to Off.

Figure 4.1 Enabled Property in the Inspector for CompleteButton



2. Select **AddButton**, labeled "Add". In the Inspector, turn the Enabled property (in the Appearance group) to Off.

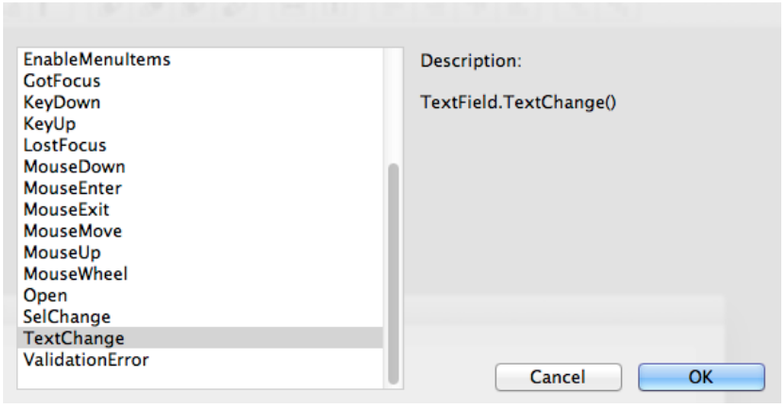
3. Select **DeleteButton**, labeled "Delete". In the Inspector, turn the Enabled property (in the Appearance group) to Off.

4. Now you will add code to enable the Add button when there is text in the Task Field.

   On the web page, double-click the **TaskField** control.

   The **Add Event Handler** window appears. Here you are seeing yet another list of event handlers. Every control type has its own specific list of event handlers. In this case, we want to disable AddButton when there is no text in the task field

Figure 4.2 Event Handlers for Text Field



and enable it when there is text. The **TextChanged** event is called whenever the text in the task field is changed, either by the user typing or by your code changing the Text property. Select **TextChanged** from the Event Handler list and click OK.

Notice the Navigator on the left updates to show the TextChange event underneath the TaskField control and the code editor displays.

5. You want to add this code:

```
If Me.Text <> "" Then
  AddButton.Enabled = True
Else
  AddButton.Enabled = False
End If
```

This code checks the Text property of the TextField (Me.Text) to see if anything is there. If there is text there, then the AddButton is enabled by setting its Enabled property to True. If there is no text, then it is disabled by setting its Enabled property to False.

6. You already added code in Chapter 3, Section 4 to prevent the Delete and Complete buttons from doing anything if no row is selected in the Task List. Now you can also make those buttons enable when a row is selected and disable when no rows are selected. This is accomplished with the ListIndex property of the Listbox.

7. Double-click the TaskList control.
   The **Add Event Handler** window appears. Here you are seeing the list of event handlers for WebListBox. The **SelectionChanged** event is called whenever the selection in the TaskList control changes.
   Select **SelectionChanged** from the Event Handler list and click OK.

8. Add this code:

```
  If Me.ListIndex >= 0 Then
    DeleteButton.Enabled = True
    CompleteButton.Enabled = True
  Else
    DeleteButton.Enabled = False
    CompleteButton.Enabled = False
  End If
```

9. Save the project by choosing File ↠ Save.

10. Run the app to test it out. Notice that the Add button is initially disabled. But try typing some text in the Task field. The Add button immediately become enabled. And if you remove the text from the Task field, the buttons again

become disabled.

Similarly, when you click on a row in the Task List, the Delete and Complete buttons become enabled.

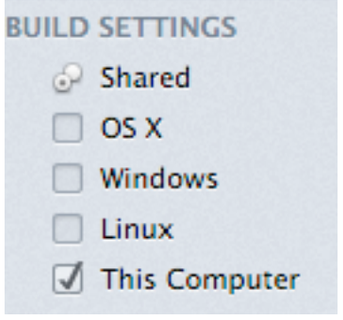
Deploying a Web Application

Sharing Your Application

Now that you have created this fine application, you probably want to share it with the world. Although you can run a web application locally and access it in your web browser, a web application is more typically run on a web server and accessed locally in your web browser. To share your web application, you want to build it and then deploy it to a web server.

Your web app can be built for any supported platform, including Windows, OS X and Linux. Most web servers use Linux, so you likely are going to build to the Linux target.

Figure 4.3 Build Settings



Deployment Options

The next thing to decide is whether you are going to deploy a Standalone web application or a CGI web application.

Standalone Web Application

A Standalone web application is an application that you manually run on your server. You have to start the application (usually from the command line) and leave it running in order for people to access the web app. In addition, a Standalone web application is accessed through a port, which you specify when building the app. Essentially, a standalone web application consists of both the web server and your web application.

Standalone web apps also take advantage of WebSockets, a feature that improves the performance of web applications by providing a faster bi-directional communications channel.

A deployed standalone web app would be accessed with a URL such as this:

http://www.mywebsite.com:8080

CGI Web Application

A web application built to use CGI uses either Apache or IIS (Microsoft Internet Information Services) as its web server. The web server then communicates to your web application using CGI. To facilitate this, a companion Perl script (supplied when you build your application) handles communication between the web server and your web application.

CGI applications cannot currently use WebSockets. Some web browsers (notably Safari) continue to display a loading indicator even after the web page has finished loading. This is a result of the method used by the web server to communicate with your web application.

Because a CGI deployment uses your existing web server software, you do not have to specify a port when accessing your web application. A typical URL looks like this:

http://www.mywebsite.com/cgi-bin/mywebapp.cgi

Deployment

Because of the wide variety of servers and their individual settings, the specifics for deploying a web app to a server is beyond the scope of this tutorial.

But in general, the steps to deploy a web application to a Linux server are simple:

1. Compile your web application for Linux.
2. Connect to your web server using FTP.
3. Upload your web application (including the Libs folder)
4. Verify that the execute flag is still set for the files that you just uploaded. Some FTP clients have been known to change this flag during upload.

The specifics can get much trickier. Refer to the Web Deployment topics in the Documentation Wiki for more details.

Section 4

All Done!

Congratulations

You have successfully completed the Web Tutorial and now have a fully functional application.

To continue on your journey of learning Xojo, you should next move on to the User Guide, which covers Xojo in its entirety.

You will also want to refer to the Language Reference, which covers the specifics of language elements, classes and other details of Xojo.