;E_ToDo_MultiAppBasics.rtfd;linkMarkername ;¬ Previous Section
;G_ToDo_ManageData.rtfd;linkMarkername ;¬ Next Section

4. To Do Tutorial

Managing Documents Through Delegation

At certain points while an application is running you want to ensure that a document's data is preserved or that a document's edited status is tracked. These events occur when users:

SquareBullet.eps ¬ Edit a document.
477373_SquareBullet.eps ¬ Close a window.
599583_SquareBullet.eps ¬ Quit the application.
708447_SquareBullet.eps ¬ Hide the application.
830902_SquareBullet.eps ¬ Switch to another application or window.

Several classes of the Application Kit send messages to their delegates when these events occur, giving the delegate the opportunity to do the appropriate thing, whether that be saving a document to the file system or marking a document as edited.

1 Mark a document as edited.

Open **ToDoDoc.m**.

Implement the **controlTextDidChange:** method to mark the document.

```
- (void)controlTextDidChange:(NSNotification *)notif
{
    [[itemMatrix window] setDocumentEdited:YES];
}
```

When a control that contains editable textÐsuch as a text field or a matrix of text fieldsÐdetects editing in a field, it posts the **controlTextDidChange:** notification which, like all notifications, is sent to the control's delegate as well as to all observers. The **setDocumentEdited:** message causes the document's window to change the image in its close button to a broken X.

TD_DocEdited.eps ¬

Note: The ToDo object that, by notification, invokes the **controlTextDidChange:** method is **itemMatrix**, the matrix of to-do items (text fields). You will programmatically set ToDoDoc to be the delegate of this object later in this tutorial.

2 Save edited documents when windows are closed.

Implement the delegation method **windowShouldClose:**.

```
- (BOOL)windowShouldClose:(id)sender
{
    int result;
/* 1 */
   if (![[itemMatrix window] isDocumentEdited]) return YES;
/* 2 */
    [[itemMatrix window] makeFirstResponder:[itemMatrix window]];
    result = NSRunAlertPanel(@"Close", @"Document has been edited.
        Save changes before closing?", @"Save", @"Don't Save",
        @"Cancel");
/* 3 */
    switch(result) {
      case NSAlertDefaultReturn: {
          [self saveDocItems];
          [self saveDoc];
          return YES;
      }
```

```
            case NSAlertAlternateReturn: {
              return YES;
            }
            case NSAlertOtherReturn: {
              return NO;
            }
        }
        return NO;
    }
```

When users click a window's close button, the window sends **windowShouldClose:** to its delegate. It expects a response directing it either to close the window or leave it open.

1. Returns YES (meaning: go ahead, close the window) if the document hasn't been edited.
2. Makes the window its own first responder. This has the effect of forcing the validation of cells, flushing currently entered text to the method that handles it (more on this in the next section).
3. Identifies the clicked button by evaluating the constant returned from **NSRunAlertPanel()** and returns the appropriate boolean value. If the user clicks the Save button, this method also updates internal storage with the currently displayed items (**saveDocItems**) and then sends **saveDoc** to itself to archive application data to a file. (**saveDocItems** is described in the following section.)

Note: Do you recall the **performClose:** method that ToDoController sends the document window when the user chooses the Close command? This method simulates a mouse click on the window's close button, causing **windowShouldClose:** to be invoked.

3 Save edited documents when the user quits the application.

In **ToDoController.m**, implement the delegation method **applicationShouldTerminate:**.

```
- (BOOL)applicationShouldTerminate:(id)sender
{
```

```
while ([NSApp keyWindow]) {
    int result;
    id doc = [[NSApp keyWindow] delegate];

    if (![[NSApp keyWindow] isDocumentEdited]) {
        [[NSApp keyWindow] close];
        if (doc) [doc autorelease];
        continue;
    }
    if ([doc isKindOfClass:[ToDoDoc class]]) {
        NSString *repfile = [[NSApp keyWindow] representedFilename];
        result = NSRunAlertPanel(@"To Do", @"Save %@?", @"Save",
            @"Don't Save", @"Cancel",
            ([repfile isEqualToString:@""]?@"UNTITLED":repfile));
        switch(result) {
          case NSAlertDefaultReturn:
            [doc saveDocItems];
            [doc saveDoc];
            break;
          case NSAlertAlternateReturn:
            [[NSApp keyWindow] close];
            break;
          case NSAlertOtherReturn:
            return NO;
        }
        if (doc) [doc autorelease];
    }
    else
        [[NSApp keyWindow] close];
}
```

```
    return YES;
}
```

NSApplication sends several message to its delegate. One of these messagesÐ**applicationShouldTerminate:**Ðnotifies the delegate that the application is about to terminate. The implementation of this method is similar to that for **windowShouldClose:**. What's different is that this method cycles through all windows of the application and, if the window is managed by ToDoDoc, puts up an attention panel and responds according to the user's choice.

Related Concepts: ;ToDoConcepts.rtfd;linkMarkername CoordinateSystemsinOpenStep;, Coordinate Systems in OpenStep

;ToDoConcepts.rtfd;linkMarkername TheApplicationQuartet:NSResponder,NSApplication,NSWindow,andNSView;, The Application Quartet: NSResponder, NSApplication, NSWindow, and NSView