Scripting ToDo List

The ToDo List application is fully scriptable – anything that you can do to a document using the keyboard and mouse, can be done by a script using the AppleEvents supported by ToDo List. Using AppleScript (which is available for free on every Macintosh), you can create powerful custom solutions that combine the features of ToDo List with other scriptable applications. For example, you might already be using ToDo List as a convenient place to keep a list of files you plan to download from Info-Mac. With a simple script, you could automatically iterate through all of the items in the list and for each, use Anarchie (which is also scriptable) to download the file to your machine.

This chapter provides an overview of the elements in ToDo List that you can control, and the commands that are available to control them. It assumes that you already have some familiarity with AppleScript.

The ToDo List Elements

Scriptable applications are typically made up of a variety of elements that can be specified (the 5th item of the document "Stuff") and a set of actions that can be performed on those elements. ToDo List provides a hierarchy of element types and a variety of ways to refer to them.

At the top of the hierarchy is a single application. The application has a number of properties (that correspond to the current preferences) that can be addressed directly:

```
autoOpen    boolean   -- should ToDo List automatically open your last documents when launched

confirmDelete    boolean   -- confirm deletes

play toggle sound    boolean   -- play sound when toggling items

saveOnClose    boolean   -- automatically save documents on close

saveInterval    integer   -- save documents after this number of minutes (0 = don't auto-save)

returnKeyClicksOK    boolean   -- <return> same as hitting OK in item dialog
```

You can retrieve or change any of these properties using AppleScript:

```
set areWeAutoOpening to the autoOpen of application "ToDo List"
set the autoOpen of application "ToDo List" to true
```

The application also contains some number of document elements (one for each ToDo List document). You can refer to the documents by name or by index:

```
document "Things To Do" of application "ToDo List"
document 3 of application "ToDo List"
```

Each document has several properties that can be addressed directly:

name string -- the title of the document

show calendar boolean -- should the calendar be shown across the top of the document

use dates boolean -- should all items have a due date associated with them

show overdue boolean -- show how many days each item is overdue

carryforward unfinished boolean -- should unfinished items be carried forward

date font string -- font to use for displaying dates

date size integer -- size of font for displaying dates

item font string -- font for displaying items

item size integer -- font size for displaying items

clipboard text only boolean -- should copying to the clipboard copy only text

voice string -- voice that should be used for speaking items

useSmallCalendarFont boolean -- if true, then a very small font is used for calendar

You can change these properties using AppleScript:

```
tell application "ToDo List"
    set the date font of document "Important Stuff" to "Times"
    set the show overdue of document 4 to false
end tell
```

Documents also contain some number of item elements. These elements can be accessed by index:

```
item 5 of document 4
```

If the list is using due dates, then items can also be accessed via the date:

```
item 3 of date "1/13/96" of document 4
item 6 of date "June 1, 1996" of document "Stuff"
```

Each item has a number of properties that you can retrieve or set:

uniqueID integer [r/o] -- a unique ID for the to do item

description string -- the text of the to do item

dueDate string -- the date the item should be done on

completed boolean -- has the item been completed (true or false)

created string [r/o] -- date and time that the item was created

priority integer -- priority of the item (0 - 10, 0 = none, 1 = highest)

modified string [r/o] -- date and time that the item was last modified

selected boolean -- true if the item is currently selected

You can access these properties from AppleScript by just fully specifying the item and property. For example:

```
tell application "ToDo List"
    -- Check off an item as finished
    set the completed of item 4 of document "Important Stuff" to true
    -- Change the due date of an item
    set the dueDate of item 2 of date "5/16/96" of
                document 3 to "May 20, 1996"
end tell
```

You can create new elements using the make command.

AppleScript Commands

ToDo List supports the standard open, print, and quit AppleEvents, so you can easily get it to launch and open a set of ToDo List documents. It also lets you save and close documents as needed:

save: save the document to disk

save document

[in alias] -- file to save the document in

close: close the document's window

close document -- document to close

[saving yes/no/ask] -- should any changes be saved

[saving in alias] -- file to save the document in

For example:

```
tell application "ToDo List"
    open alias "My Harddisk:Lists:Stuff To Do"
    set the voice of document "Stuff To Do" to "Victoria"
    save document "Stuff To Do"
    close document "Stuff To Do"
end tell
```

ToDo List also supports several core AppleEvents that let you count, create, delete, etc. the various elements that it defines:

count: count the number of elements in a class

count reference -- the object whose elements are to be counted

each type class -- the class of the elements to be counted.

Result: integer -- the number of elements in the given class

For example:

```
tell application "ToDo List"
    set cnt to count of documents
    set numItems to count of item of document 3
    set dayItems to count each item of date "5/6/96" of document "Stuff"
    set numDates to count each date of document 1
end tell
```

get: get data for a to do item

get reference -- reference to the item to get

Result: anything -- data for the requested to do item

set: set an item or property of an item to some value

set reference -- the item or item property to change

to anything -- new value for the item or property

make: make a new item

make

new type class

[at integer]

with properties record

within reference -- the document to which the new item should be added

Result: integer -- index of the new item

Makes a new item of the given type ("item" or "document"). Any unspecified properties default to the same values as they would if the item were created directly in ToDo List. For example:

```
tell application "ToDo List"
    -- Note: make new document doesn't support 'with properties'!
    set docNum to make new document
    set docName to get name of document docNum
    set the use dates of document docName to true
    make new item at 1 within document docName
        with properties  description:"testing", priority: 1,
                         dueDate:"Feb 1, 1996"
    make new item at 9999 within document docName
        with properties  description:"testing", priority: 0,
                         completed: true, dueDate:"Feb 1, 1996"
end tell
```

delete: delete an item from the document

delete reference -- the item to delete

For example:

```
tell application "ToDo List"
    delete item 1 of document "Stuff"
    delete item 1 of date "5/5/96" of document 2
end tell
```

move: move the referenced item to a new location, changing its due date and priority if necessary

move reference -- a reference to the item to be moved

to integer -- index of the location the item should be moved to

Moves an item from one location to anotherwithin the same document. If necessary, the priority or due date of the item will be modified to match the new location. For example:

```
tell application "ToDo List"
    -- Move the first item to the end of the list
    move item 1 of document "Stuff" to 9999
    -- Swap the first and second items
    move item 1 of document "Stuff" to 2
end tell
```

ToDo List also supports several custom AppleEvents that let you control the special features of the application:

SpeakItems: Read all of the unfinished items in the list

SpeakItems document -- the ToDo List document to speak items from

SpeakTodaysItems: Read all of the unfinished items for today

SpeakTodaysItems document -- the ToDo List document to read today's items from

synchronizeNewton: synchronize the given document with a Newton

synchronizeNewton document -- the ToDo List document to synchronize to

[via serial-modem/serial-printer/modem/AppleTalk]

-- synchronize Newton using this connection (serial, AppleTalk, modem)

[using string] -- option string for the connection

export: export the selected items (or all if none are selected) in the given ToDo List document to a tab-delimited text file.

export document -- the ToDo List document to export to a text file

[as text/HTML] -- type of document to create (text or HTML).

[in alias] -- the file to export the ToDo List document into

import: import items from a tab-delimited text file

import document -- the ToDo List document to import the items into

[from alias] -- the tab-delimited text file to import the items from