Things AppleScript Guide

Revision 14, 2013-02-01

http://www.culturedcode.com/

The basics 4
Lists 4

Working with to-dos 6
Getting to-dos 6
Getting to-dos in a specific list 6
Getting project to-dos 6
Getting to-dos in a specific area 7
Adding new to-dos 7
Setting properties for a to do 9

Working with projects 10
Getting projects 10
Adding new projects 10
Setting properties for a project 11

Working with areas 12
Getting areas 12
Adding new areas 12
Setting properties for an area 12
Deleting areas 13

Working with contacts 14
Getting contacts 14
Adding new contacts 14
Getting all to-dos and projects assigned to a contact 14
Assigning items to contacts 15
Canceling the assignment of contacts 15

**Moving items around** 16
Moving to-dos/projects between lists 16
Exception: scheduling to-dos and projects 16
Marking to-dos and projects as completed 17
Marking to-dos and projects as canceled 17
Assigning projects to to-dos 17
Assigning areas to projects/to-dos 18
Detaching to-dos/projects from projects/areas 18
Deleting to-dos and projects 18

**Working with tags** 20
Getting all available tags 20
Creating new tags 20
Getting to do/project tags 20
Setting tags 21
Working with tag hierarchies 21
Deleting tags 22

**User interface interactions** 23
Getting current selected to-dos 23
Selecting a focus, project, area, or to do 23
Editing a to do/project 23

**Integration** 25
Displaying the quick entry panel 25
Other actions 26
Emptying the trash 26
Logging completed items 26

Example scripts 27
Importing items from a text file 27
The basics

Each predefined list visible in the left panel of Things can be accessed via AppleScript.

Lists

You can access the following lists of Things using their names:

- list “Inbox”
- list “Today”
- list “Next”
- list “Scheduled”
- list “Someday”
- list “Projects”
- list “Logbook”
- list “Trash”

They are predefined and you may not create any new lists using the `make` command.

Each list contains collections of to-dos.

```applescript
tell application "Things"
    set inboxToDos to to dos of list "Inbox"
    repeat with inboxToDo in inboxToDos
        --- do something with each inbox to do using inboxToDo variable
    end repeat
end tell
```

Please note: the sort order of to-dos and projects accessed using “to dos” and “projects” collections corresponds to the UI. For instance:

```applescript
tell application "Things"
    set scheduledToDos to to dos of list "Scheduled"
    repeat with scheduledToDo in scheduledToDos
        --- to-dos get sorted by schedule date - just like in Things UI
    end repeat
end tell
```
Working with to-dos

Getting to-dos

You can access all to-dos (including projects) defined in Things using the application-level collection named “to dos”. You may also use the to do’s name:

```
tell application "Things"
    repeat with todo in to dos
        --- do something with each to do using todo variable
    end repeat
    set todo_things to to do named "Things"
end tell
```

Getting to-dos in a specific list

Please use the collection “to dos” of a given list:

```
tell application "Things"
    repeat with inboxToDo in to dos of list "Inbox"
        --- do something with each to do using ToDo variable
    end repeat
end tell
```

Getting project to-dos

You can access all project to-dos using the project’s collection named “to dos”:
Getting to-dos in a specific area

Please use the collection “to dos” of a given area:

```
tell application "Things"
    repeat with ccToDo in to dos of area "Cultured Code"
        --- do something with each to do using ccToDo variable
    end repeat
end tell
```

Adding new to-dos

Typically, you’ll create new items using the standard *make* AppleScript command.

You may use the simple syntax to add a new Inbox to do:

```
tell application "Things"
    set newToDo to make new to do ¬
        with properties {name:"New to do", due date:current date}
end tell
```

You can also specify the container for the new to do immediately. It can be a list, a project, an area, or a contact:
tell application "Things"

-- adding to dos in lists

set newToDo to make new to do ¬
    with properties {name:"New to do", due date:current date} ¬
    at beginning of list "Today"

tell list "Someday"
    set newToDo to make new to do ¬
        with properties {name:"New to do for someday"}
end tell

-- adding to dos in projects

set newToDo to make new to do ¬
    with properties {name:"New Things feature", due date:current date} ¬
    at beginning of project "Things"

tell project "Things"
    set newToDo to make new to do ¬
        with properties {name:"New Things feature"}
end tell

-- adding to dos in areas

set newToDo to make new to do ¬
    with properties {name:"New personal to do"} ¬
    at beginning of area "Personal"

tell area "Personal"
    set newToDo to make new to do ¬
        with properties {name:"New personal to do"}
end tell

-- adding delegated to dos

set newToDo to make new to do ¬
    with properties {name:"New delegated to do"} ¬
    at beginning of contact "Steve Jobs"

tell contact "Steve Jobs"
    set newToDo to make new to do ¬
        with properties {name:"New delegated to do"}
end tell
end tell
Setting properties for a to do

Here’s how you can set the properties of a task:

```
tell application "Things"
    set newToDo to make new to do ¬
        with properties {name:"New to do", due date:current date} ¬
        at beginning of list "Next"

    set name of newToDo to "This task has been renamed"
    set notes of newToDo to "http://www.cultured.com/" & linefeed & "call Werner for more details"
    set due date of newToDo to (current date) + 7 * days
    set completion date of newToDo to current date
    set tag names of newToDo to "Home, Mac"
```

Things AppleScript Guide
Working with projects

Getting projects

You can access all projects defined in Things using the application-level collection named “projects”. You may also use the project’s name:

```appscript
tell application "Things"
    repeat with pr in projects
        --- do something with each project using pr variable
    end repeat
    set pr_things to project "Things"
end tell
```

Alternatively, you may also use the to dos collection of “Projects” list:

```appscript
tell application "Things"
    repeat with pr in to dos of list "Projects"
        --- do something with each project using pr variable
    end repeat
end tell
```

Adding new projects

Please use the standard `make` AppleScript command. You can use the simple syntax:

```appscript
tell application "Things"
    set newProject to make new project ¬
        with properties {name:"New project", notes:"Tiny note"}
end tell
```

...or specify the container - “Projects” list:
**Setting properties for a project**

Here’s how you can set the properties of a project:

```
tell application "Things"
    set newProject to make new project ¬
        with properties {name:"New project", notes:"Tiny note"} ¬
    at beginning of list "Projects"

    -- or

tell list "Projects"
    set newProject to make new project ¬
        with properties {name:"New project", notes:"Tiny note"}
    end tell
end tell
```

```
tell application "Things"
    set newProject to make new project ¬
        with properties {name:"New project"} ¬
    at beginning of list "Projects"

    set name of newProject to "This project has been renamed"
    set notes of newProject to "http://www.cultured.com/" & linefeed & "call Werner for more details"
    set due date of newProject to (current date) + 7 * days
    set completion date of newProject to current date
    set tag names of newProject to "Home, Mac"
end tell
```
Working with areas

Getting areas

Areas are accessible using the application-level collection named “areas”. You may also use the name of the given area:

```applescript
tell application "Things"
    repeat with ar in areas
        --- do something with the area using ar variable
    end repeat
    set ar_things to area "Private"
end tell
```

Adding new areas

Please use the standard `make` AppleScript command:

```applescript
tell application "Things"
    set newArea to make new area ¬
        with properties {name:"New area", suspended:false}
end tell
```

Setting properties for an area

Here’s how you can set the properties of an area:
Deleting areas

Please use the standard `delete` AppleScript command. It will move the area to Trash.

tell application "Things"
    set anArea to area named "Area to remove"
    delete anArea
end tell
Working with contacts

Getting contacts

Your Things contacts are accessible using the application-level collection named “contacts”. You may also use the contact's display name:

```app
tell application "Things"
    repeat with aContact in contacts
        --- do something with the contact using the aContact variable
    end repeat
    set thatsMe to contact "Bartlomiej Bargiel"
end tell
```

Adding new contacts

New Things contacts have to be assigned to a given contact from your Address Book right at the moment of their creation. That’s why you’re supposed to use a dedicated command “add contact named” with the Address Book name as parameter:

```app
tell application "Things"
    set newContact to add contact named "Bartlomiej Bargiel"
end tell
```

Things will perform an AddressBook search and create a proper contact record corresponding to the address card of Bartlomiej Bargiel from the Address Book.

Getting all to-dos and projects assigned to a contact

Each contact object contains collections of assigned to-dos. Here’s how to use them:
Assigning items to contacts

Please use the "contact" property of to-dos/projects to assign them to a contact:

```applescript
tell application "Things"
    set aContact to contact named "My friend"
    repeat with aToDo in to dos of aContact
        --- do something with each to do assigned to my friend
    end repeat
end tell
```

Canceling the assignment of contacts

Please use the `delete` command to remove the contact from todos/projects:

```applescript
tell application "Things"
    set aContact to contact named "Somebody Else"
    repeat with aToDo in to dos of list "Today"
        set contact of aToDo to aContact
    end repeat
end tell
```
Moving items around

Moving to-dos/projects between lists

The most common operation you perform using the Things UI is moving your activities between lists.

This can be done using the “move” command and works for all to-dos and projects. You’ll only need to specify the target list to move the to do/project to:

```applescript
    tell application "Things"
        set newToDo to make new to do ¬
            with properties {name:"New to do for today"} at beginning of list "Inbox"
        move newToDo to list "Today"
        --- this marks the project as completed
        set pr to project "Things"
        move pr to list "Logbook"
    end tell
```

Exception: scheduling to-dos and projects

Moving items to “Scheduled” list is not supported - Things requires all scheduled actions to get an activation date.

Please use the `schedule` command to schedule your to dos and projects:

```applescript
    tell application "Things"
        set toDoToSchedule to first to do of list "Inbox"
        schedule toDoToSchedule for (current date) + 1 * days
    end tell
```

The above script will set the activation date to tomorrow and put the to do to Scheduled list immediately.
Marking to-dos and projects as completed

You can complete a to do/project by moving it to the Logbook list:

```appscript
tell application "Things"
    set pr to project "Things"
    move pr to list "Logbook"
end tell
```

but you may also want it to stay in its current list before it gets logged (see: `log completed now` command) - please use the `status` property of to-dos and projects then:

```appscript
tell application "Things"
    set toDoToComplete to to do named "To do to complete"
    set status of toDoToComplete to completed
end tell
```

Marking to-dos and projects as canceled

Moving a to do to logbook is not enough to mark it as canceled. Please use the `status` property of a given to do to cancel it:

```appscript
tell application "Things"
    set toDoToCancel to to do named "To do to cancel"
    set status of toDoToCancel to canceled
end tell
```

Assigning projects to to-dos

You can get or set the to-do’s project using the “project” property:
Assigning areas to projects/to-dos

You can get or set the to do/project's area using the “area” property:

```applescript
tell application "Things"
    set aToDo to first to do
    set project of aToDo to project "Things"
end tell
```

Detaching to-dos/projects from projects/areas

You can detach an item from its project or area using the standard AppleScript command “delete”:

```applescript
--- this will detach the to do from its project
set aToDo to first to do of project named "Things"
delete project of aToDo

--- this will detach the project from its area
set aProject to project "Things"
delete area of aProject
end tell
```

Deleting to-dos and projects
The best location for deleted items is: Trash! You can simply use the “move” command to move the items to the Things Trash list:

```appleScript
tell application "Things"
    set aToDo to first to do
    move aToDo to list "Trash"
end tell
```

You can use the standard `delete` AppleScript command, too:

```appleScript
tell application "Things"
    set aToDo to first to do
    delete aToDo
    set aProject to first project
    delete aProject
end tell
```
Working with tags

Getting all available tags

Application object supports the “tags” collection which you can use to access all tags defined in Things:

```
tell application "Things"
    repeat with aTag in tags
        --- do something with each tag using the aTag variable
    end repeat
end tell
```

Creating new tags

Please use the standard *make* AppleScript command:

```
tell application "Things"
    set newTag to make new tag with properties {name:"Home"}
end tell
```

If a tag with the given name already exists, it will be returned.

Getting to do/project tags

You can use *string* or *tag object* accessors to access the tags of Things items:
Setting tags

The most convenient way of setting tags for to-dos and projects is to use the “tag names” property which accepts a comma-separated string of tag names:

```applescript
tell application "Things"
    set aToDo to first to do of list "Inbox"
    --- output: "Home, Mac" - a string!
    set tagNames to tag names of aToDo
    --- output: a list of tag objects
    set tagList to tags of aToDo
    repeat with aTag in tagList
        --- do something with each tag
    end repeat
end tell
```

Working with tag hierarchies

You can use the parent tag property to set a parent for a given tag:

```applescript
tell application "Things"
    set homeTag to tag "Home"
    set parent tag of homeTag to tag "Places"
end tell
```
Deleting tags

Please use the standard `delete` AppleScript command:

```applescript
tell application "Things"
    set aTag to tag "to_remove"
    delete aTag
end tell
```
User interface interactions

Getting current selected to-dos

You can access all selected to-dos using the application-level collection named “selected to dos”:

```app
tell application "Things"
  repeat with selectedToDo in selected to dos
    move selectedToDo to list "Today"
  end repeat
end tell
```

Selecting a focus, project, area, or to do

In order to display any list, to do, project, area, or contact in the Things UI, please use the `show` command:

```app
tell application "Things"
  -- switches to the proper list and selects the given to do in the list of to-dos
  show to do "Write AppleScript Guide"

  -- selects the "Things" project in the left Things panel and shows its contents
  show project "Things"
end tell
```

Editing a to do/project

When adding a new to do or project, you may wish to let the user edit its details manually. Please use the `edit` command in that case:
tell application "Things"

    set newTaskToEdit to make new to do ¬
        with properties {name:"Please enter name"} ¬
        at beginning of list "Today"

    -- Things will switch to "Today" list and enter the edit mode
    edit newTaskToEdit

    -- Same with projects
    set newProjectToEdit to make new project ¬
        with properties {name:"Please enter name"} ¬
        at beginning of list "Projects"
    edit newProjectToEdit

end tell
Integration

Displaying the quick entry panel

Please use the *show quick entry panel* command to display the Quick Entry panel.

```applescript
tell application "Things"
    show quick entry panel
end tell
```

An optional *with properties* parameter lets you fill the panel with given values:

```applescript
tell application "Things"
    show quick entry panel with properties ¬
    {name:"Enter a title for this to do", notes:"Enter a note"}
end tell
```

Instead of the *with properties* parameter, you can also specify *with autofill*, which will try to pre-populate the panel with information from the current application the user is running:

```applescript
tell application "Things"
    show quick entry panel with autofill yes
end tell
```
Other actions

Emptying the trash

Please use the “empty trash” command:

```
tell application "Things"
    empty trash
end tell
```

Logging completed items

Please use the “log completed now” command:

```
tell application "Things"
    log completed now
end tell
```
Example scripts

Importing items from a text file

```
set myFile to (choose file with prompt "Select a file to read:")
open for access myFile

set fileContents to read myFile using delimiter {linefeed}
close access myFile

tell application "Things"

    repeat with currentLine in reverse of fileContents
        set newToDo to make new to do ¬
            with properties {name:currentLine} ¬
            at beginning of list "Next"
        -- perform some other operations using newToDo
    end repeat

end tell
```