Inhaltsverzeichnis

Modul:Message box/Doku

Dies ist die Dokumentationsseite für Modul:Message box

Vorlage:Lua

This is a meta-module that implements the message box templates Vorlage:Tl, Vorlage:Tl, Vorlage:Tl, Vorlage:Tl, Vorlage:Tl, Vorlage:Tl, and Vorlage:Tl. It is intended to be used from Lua modules, and should not be used directly from wiki pages. If you want to use this module's functionality from a wiki page, please use the individual message box templates instead.

Usage

To use this module from another Lua module, first you need to load it.

```
local messageBox = require('Module:Message box')
```

To create a message box, use the `main` function. It takes two parameters: the first is the box type (as a string), and the second is a table containing the message box parameters.

```
local box = messageBox.main( boxType, {
    param1 = param1,
    param2 = param2,
    -- More parameters...
})
```

There are seven available box types:

Box type	Template	Purpose
mbox	Vorlage:Tl	For message boxes to be used in multiple namespaces
ambox	Vorlage:Tl	For article message boxes
cmbox	Vorlage:Tl	For category message boxes
fmbox	Vorlage:Tl	For interface message boxes
imbox	Vorlage:Tl	For file namespace message boxes
tmbox	Vorlage:Tl	For talk page message boxes
ombox	Vorlage:Tl	For message boxes in other namespaces

See the template page of each box type for the available parameters.

Usage from #invoke

As well as the `main` function, this module has separate functions for each box type. They are accessed using the code `{{#invoke:Message box|mbox|...}}`, `{{#invoke:Message box|ambox|...}}`, etc. These will work when called from other modules, but they access code used to process arguments passed from #invoke, and so calling them will be less efficient than calling `main`.

Technical details

The module uses the same basic code for each of the templates listed above; the differences between each of them are configured using the data at Module:Message box/configuration. Here are the various configuration options and what they mean:

- `types` – a table containing data used by the type parameter of the message box. The table keys are the values that can be passed to the type parameter, and the table values are tables containing the class and the image used by that type.
- `default` – the type to use if no value was passed to the type parameter, or if an invalid value was specified.
- `showInvalidTypeError` – whether to show an error if the value passed to the type parameter was invalid.
- `allowBlankParams` – usually blank values are stripped from parameters passed to the module. However, whitespace is preserved for the parameters included in the allowBlankParams table.
- `allowSmall` – whether a small version of the message box can be produced with "small=yes".
- `smallParam` – a custom name for the small parameter. For example, if set to "left" you can produce a small message box using "small=left".
- `smallClass` – the class to use for small message boxes.
- `substCheck` – whether to perform a subst check or not.
- `classes` – an array of classes to use with the message box.
- `imageEmptyCell` – whether to use an empty Vorlage:Tag cell if there is no image set. This is used to preserve spacing for message boxes with a width of less than 100% of the screen.
- `imageEmptyCellStyle` – whether empty image cells should be styled.
- `imageCheckBlank` – whether "image=blank" results in no image being displayed.
- `imageSmallSize` – usually, images used in small message boxes are set to 30x30px. This sets a custom size.
- `imageCellDiv` – whether to enclose the image in a div enforcing a maximum image size.
- `useCollapsibleTextFields` – whether to use text fields that can be collapsed, i.e. "issue", "fix", "talk", etc. Currently only used in ambox.
- `imageRightNone` – whether imageright=none results in no image being displayed on the right-hand side of the message box.
- `sectionDefault` – the default name for the "section" parameter. Depends on `useCollapsibleTextFields`.
- `allowMainspaceCategories` – allow categorisation in the main namespace.
- `templateCategory` – the name of a category to be placed on the template page.

- `templateCategoryRequireName` – whether the `name` parameter is required to display the template category.
- `templateErrorCategory` – the name of the error category to be used on the template page.
- `templateErrorParamsToCheck` – an array of parameter names to check. If any are absent, the `templateErrorCategory` is applied to the template page.

Modul:Message box

Vorlage:Lua

This is a meta-module that implements the message box templates Vorlage:Tl, Vorlage:Tl, Vorlage:Tl, Vorlage:Tl, Vorlage:Tl, Vorlage:Tl, and Vorlage:Tl. It is intended to be used from Lua modules, and should not be used directly from wiki pages. If you want to use this module's functionality from a wiki page, please use the individual message box templates instead.

Usage

To use this module from another Lua module, first you need to load it.

```
local messageBox = require('Module:Message box')
```

To create a message box, use the `main` function. It takes two parameters: the first is the box type (as a string), and the second is a table containing the message box parameters.

```
local box = messageBox.main( boxType, {
    param1 = param1,
    param2 = param2,
    -- More parameters...
})
```

There are seven available box types:

Box type	Template	Purpose
mbox	Vorlage:Tl	For message boxes to be used in multiple namespaces
ambox	Vorlage:Tl	For article message boxes
cmbox	Vorlage:Tl	For category message boxes
fmbox	Vorlage:Tl	For interface message boxes
imbox	Vorlage:Tl	For file namespace message boxes
tmbox	Vorlage:Tl	For talk page message boxes
ombox	Vorlage:Tl	For message boxes in other namespaces

See the template page of each box type for the available parameters.

Usage from #invoke

As well as the `main` function, this module has separate functions for each box type. They are accessed using the code `{{#invoke:Message box|mbox|...}}`, `{{#invoke:Message box|ambox|...}}`, etc. These will work when called from other modules, but they access code used to process arguments passed from #invoke, and so calling them will be less efficient than calling `main`.

Technical details

The module uses the same basic code for each of the templates listed above; the differences between each of them are configured using the data at Module:Message box/configuration. Here are the various configuration options and what they mean:

- `types` – a table containing data used by the type parameter of the message box. The table keys are the values that can be passed to the type parameter, and the table values are tables containing the class and the image used by that type.
- `default` – the type to use if no value was passed to the type parameter, or if an invalid value was specified.
- `showInvalidTypeError` – whether to show an error if the value passed to the type parameter was invalid.
- `allowBlankParams` – usually blank values are stripped from parameters passed to the module. However, whitespace is preserved for the parameters included in the allowBlankParams table.
- `allowSmall` – whether a small version of the message box can be produced with "small=yes".
- `smallParam` – a custom name for the small parameter. For example, if set to "left" you can produce a small message box using "small=left".
- `smallClass` – the class to use for small message boxes.
- `substCheck` – whether to perform a subst check or not.
- `classes` – an array of classes to use with the message box.
- `imageEmptyCell` – whether to use an empty Vorlage:Tag cell if there is no image set. This is used to preserve spacing for message boxes with a width of less than 100% of the screen.
- `imageEmptyCellStyle` – whether empty image cells should be styled.
- `imageCheckBlank` – whether "image=blank" results in no image being displayed.
- `imageSmallSize` – usually, images used in small message boxes are set to 30x30px. This sets a custom size.
- `imageCellDiv` – whether to enclose the image in a div enforcing a maximum image size.
- `useCollapsibleTextFields` – whether to use text fields that can be collapsed, i.e. "issue", "fix", "talk", etc. Currently only used in ambox.
- `imageRightNone` – whether imageright=none results in no image being displayed on the right-hand side of the message box.
- `sectionDefault` – the default name for the "section" parameter. Depends on useCollapsibleTextFields.
- `allowMainspaceCategories` – allow categorisation in the main namespace.
- `templateCategory` – the name of a category to be placed on the template page.

- `templateCategoryRequireName` – whether the `name` parameter is required to display the template category.
- `templateErrorCategory` – the name of the error category to be used on the template page.
- `templateErrorParamsToCheck` – an array of parameter names to check. If any are absent, the `templateErrorCategory` is applied to the template page.

```lua
-- This is a meta-module for producing message box templates, including
-- {{mbox}}, {{ambox}}, {{imbox}}, {{tmbox}}, {{ombox}}, {{cmbox}} and {{fmbox}}

-- Load necessary modules.
require('Module:No globals')
local getArgs
local yesno = require('Module:Yesno')

-- Get a language object for formatDate and ucfirst.
local lang = mw.language.getContentLanguage()

-- Define constants
local CONFIG_MODULE = 'Module:Message box/configuration'
local DEMOSPACES = {talk = 'tmbox', image = 'imbox', file = 'imbox', category =

--------------------------------------------------------------------------------
-- Helper functions
--------------------------------------------------------------------------------

local function getTitleObject(...)
        -- Get the title object, passing the function through pcall
        -- in case we are over the expensive function count limit.
        local success, title = pcall(mw.title.new, ...)
        if success then
                return title
        end
end

local function union(t1, t2)
        -- Returns the union of two arrays.
        local vals = {}
        for i, v in ipairs(t1) do
                vals[v] = true
        end
        for i, v in ipairs(t2) do
                vals[v] = true
        end
        local ret = {}
        for k in pairs(vals) do
                table.insert(ret, k)
        end
        table.sort(ret)
        return ret
end

local function getArgNums(args, prefix)
        local nums = {}
        for k, v in pairs(args) do
                local num = mw.ustring.match(tostring(k), '^' .. prefix .. '([1-9
                if num then
                        table.insert(nums, tonumber(num))
                end
        end
        table.sort(nums)
```

```
        return nums
end

-------------------------------------------------------------------------
-- Box class definition
-------------------------------------------------------------------------

local MessageBox = {}
MessageBox.__index = MessageBox

function MessageBox.new(boxType, args, cfg)
        args = args or {}
        local obj = {}

        -- Set the title object and the namespace.
        obj.title = getTitleObject(args.page) or mw.title.getCurrentTitle()

        -- Set the config for our box type.
        obj.cfg = cfg[boxType]
        if not obj.cfg then
                local ns = obj.title.namespace
                -- boxType is "mbox" or invalid input
                if args.demospace and args.demospace ~= '' then
                        -- implement demospace parameter of mbox
                        local demospace = string.lower(args.demospace)
                        if DEMOSPACES[demospace] then
                                -- use template from DEMOSPACES
                                obj.cfg = cfg[DEMOSPACES[demospace]]
                        elseif string.find( demospace, 'talk' ) then
                                -- demo as a talk page
                                obj.cfg = cfg.tmbox
                        else
                                -- default to ombox
                                obj.cfg = cfg.ombox
                        end
                elseif ns == 0 then
                        obj.cfg = cfg.ambox -- main namespace
                elseif ns == 6 then
                        obj.cfg = cfg.imbox -- file namespace
                elseif ns == 14 then
                        obj.cfg = cfg.cmbox -- category namespace
                else
                        local nsTable = mw.site.namespaces[ns]
                        if nsTable and nsTable.isTalk then
                                obj.cfg = cfg.tmbox -- any talk namespace
                        else
                                obj.cfg = cfg.ombox -- other namespaces or invali
                        end
                end
        end

        -- Set the arguments, and remove all blank arguments except for the ones
        -- listed in cfg.allowBlankParams.
        do
                local newArgs = {}
                for k, v in pairs(args) do
                        if v ~= '' then
                                newArgs[k] = v
                        end
                end
                for i, param in ipairs(obj.cfg.allowBlankParams or {}) do
                        newArgs[param] = args[param]
                end
                obj.args = newArgs
```

```lua
        end

        -- Define internal data structure.
        obj.categories = {}
        obj.classes = {}
        -- For lazy loading of [[Module:Category handler]].
        obj.hasCategories = false

        return setmetatable(obj, MessageBox)
end

function MessageBox:addCat(ns, cat, sort)
        if not cat then
                return nil
        end
        if sort then
                cat = string.format('[[Category:%s|%s]]', cat, sort)
        else
                cat = string.format('[[Category:%s]]', cat)
        end
        self.hasCategories = true
        self.categories[ns] = self.categories[ns] or {}
        table.insert(self.categories[ns], cat)
end

function MessageBox:addClass(class)
        if not class then
                return nil
        end
        table.insert(self.classes, class)
end

function MessageBox:setParameters()
        local args = self.args
        local cfg = self.cfg

        -- Get type data.
        self.type = args.type
        local typeData = cfg.types[self.type]
        self.invalidTypeError = cfg.showInvalidTypeError
                and self.type
                and not typeData
        typeData = typeData or cfg.types[cfg.default]
        self.typeClass = typeData.class
        self.typeImage = typeData.image

        -- Find if the box has been wrongly substituted.
        self.isSubstituted = cfg.substCheck and args.subst == 'SUBST'

        -- Find whether we are using a small message box.
        self.isSmall = cfg.allowSmall and (
                cfg.smallParam and args.small == cfg.smallParam
                or not cfg.smallParam and yesno(args.small)
        )

        -- Add attributes, classes and styles.
        self.id = args.id
        self.name = args.name
        if self.name then
                self:addClass('box-' .. string.gsub(self.name,' ','_'))
        end
        if yesno(args.plainlinks) ~= false then
                self:addClass('plainlinks')
        end
```

```
        for _, class in ipairs(cfg.classes or {}) do
                self:addClass(class)
        end
        if self.isSmall then
                self:addClass(cfg.smallClass or 'mbox-small')
        end
        self:addClass(self.typeClass)
        self:addClass(args.class)
        self.style = args.style
        self.attrs = args.attrs

        -- Set text style.
        self.textstyle = args.textstyle

        -- Find if we are on the template page or not. This functionality is only
        -- used if useCollapsibleTextFields is set, or if both cfg.templateCatego
        -- and cfg.templateCategoryRequireName are set.
        self.useCollapsibleTextFields = cfg.useCollapsibleTextFields
        if self.useCollapsibleTextFields
                or cfg.templateCategory
                and cfg.templateCategoryRequireName
        then
                if self.name then
                        local templateName = mw.ustring.match(
                                self.name,
                                '^[tT][eE][mM][pP][lL][aA][tT][eE][%s_]*:[%s_]*(
                        ) or self.name
                        templateName = 'Template:' .. templateName
                        self.templateTitle = getTitleObject(templateName)
                end
                self.isTemplatePage = self.templateTitle
                        and mw.title.equals(self.title, self.templateTitle)
        end

        -- Process data for collapsible text fields. At the moment these are onl
        -- used in {{ambox}}.
        if self.useCollapsibleTextFields then
                -- Get the self.issue value.
                if self.isSmall and args.smalltext then
                        self.issue = args.smalltext
                else
                        local sect
                        if args.sect == '' then
                                sect = 'This ' .. (cfg.sectionDefault or 'page')
                        elseif type(args.sect) == 'string' then
                                sect = 'This ' .. args.sect
                        end
                        local issue = args.issue
                        issue = type(issue) == 'string' and issue ~= '' and issue
                        local text = args.text
                        text = type(text) == 'string' and text or nil
                        local issues = {}
                        table.insert(issues, sect)
                        table.insert(issues, issue)
                        table.insert(issues, text)
                        self.issue = table.concat(issues, ' ')
                end

                -- Get the self.talk value.
                local talk = args.talk
                -- Show talk links on the template page or template subpages if t
                -- parameter is blank.
                if talk == ''
                        and self.templateTitle
```

```
                and (
                        mw.title.equals(self.templateTitle, self.title)
                        or self.title:isSubpageOf(self.templateTitle)
                )
        then
                talk = '#'
        elseif talk == '' then
                talk = nil
        end
        if talk then
                -- If the talk value is a talk page, make a link to that
                -- assume that it's a section heading, and make a link to
                -- page of the current page with that section heading.
                local talkTitle = getTitleObject(talk)
                local talkArgIsTalkPage = true
                if not talkTitle or not talkTitle.isTalkPage then
                        talkArgIsTalkPage = false
                        talkTitle = getTitleObject(
                                self.title.text,
                                mw.site.namespaces[self.title.namespace]
                        )
                end
                if talkTitle and talkTitle.exists then
local talkText
if self.isSmall then
    local talkLink = talkArgIsTalkPage and talk or (talkTitle.pre
    talkText = string.format('([[%s|talk]])', talkLink)
else
    talkText = 'Relevant discussion may be found on'
    if talkArgIsTalkPage then
        talkText = string.format(
            '%s [[%s|%s]].',
            talkText,
            talk,
            talkTitle.prefixedText
        )
    else
        talkText = string.format(
            '%s the [[%s#%s|talk page]].',
            talkText,
            talkTitle.prefixedText,
            talk
        )
    end
end
                self.talk = talkText
                end
        end

        -- Get other values.
        self.fix = args.fix ~= '' and args.fix or nil
        local date
        if args.date and args.date ~= '' then
                date = args.date
        elseif args.date == '' and self.isTemplatePage then
                date = lang:formatDate('F Y')
        end
        if date then
                self.date = string.format(" <span class='date-container'>
        end
        self.info = args.info
        if yesno(args.removalnotice) then
                self.removalNotice = cfg.removalNotice
        end
```

```lua
		end

		-- Set the non-collapsible text field. At the moment this is used by all
		-- types other than ambox, and also by ambox when small=yes.
		if self.isSmall then
			self.text = args.smalltext or args.text
		else
			self.text = args.text
		end

		-- Set the below row.
		self.below = cfg.below and args.below

		-- General image settings.
		self.imageCellDiv = not self.isSmall and cfg.imageCellDiv
		self.imageEmptyCell = cfg.imageEmptyCell
		if cfg.imageEmptyCellStyle then
			self.imageEmptyCellStyle = 'border:none;padding:0;width:1px'
		end

		-- Left image settings.
		local imageLeft = self.isSmall and args.smallimage or args.image
		if cfg.imageCheckBlank and imageLeft ~= 'blank' and imageLeft ~= 'none'
			or not cfg.imageCheckBlank and imageLeft ~= 'none'
		then
			self.imageLeft = imageLeft
			if not imageLeft then
				local imageSize = self.isSmall
					and (cfg.imageSmallSize or '30x30px')
					or '40x40px'
				self.imageLeft = string.format('[[File:%s|%s|link=|alt=]]
					or 'Imbox notice.png', imageSize)
			end
		end

		-- Right image settings.
		local imageRight = self.isSmall and args.smallimageright or args.imagerig
		if not (cfg.imageRightNone and imageRight == 'none') then
			self.imageRight = imageRight
		end
end

function MessageBox:setMainspaceCategories()
	local args = self.args
	local cfg = self.cfg

	if not cfg.allowMainspaceCategories then
		return nil
	end

	local nums = {}
	for _, prefix in ipairs{'cat', 'category', 'all'} do
		args[prefix .. '1'] = args[prefix]
		nums = union(nums, getArgNums(args, prefix))
	end

	-- The following is roughly equivalent to the old {{Ambox/category}}.
	local date = args.date
	date = type(date) == 'string' and date
	local preposition = 'from'
	for _, num in ipairs(nums) do
		local mainCat = args['cat' .. tostring(num)]
			or args['category' .. tostring(num)]
		local allCat = args['all' .. tostring(num)]
```

```
                    mainCat = type(mainCat) == 'string' and mainCat
                    allCat = type(allCat) == 'string' and allCat
                    if mainCat and date and date ~= '' then
                            local catTitle = string.format('%s %s %s', mainCat, prepo
                            self:addCat(0, catTitle)
                            catTitle = getTitleObject('Category:' .. catTitle)
                            if not catTitle or not catTitle.exists then
                                    self:addCat(0, 'Articles with invalid date parame
                            end
                    elseif mainCat and (not date or date == '') then
                            self:addCat(0, mainCat)
                    end
                    if allCat then
                            self:addCat(0, allCat)
                    end
            end
    end

    function MessageBox:setTemplateCategories()
            local args = self.args
            local cfg = self.cfg

            -- Add template categories.
            if cfg.templateCategory then
                    if cfg.templateCategoryRequireName then
                            if self.isTemplatePage then
                                    self:addCat(10, cfg.templateCategory)
                            end
                    elseif not self.title.isSubpage then
                            self:addCat(10, cfg.templateCategory)
                    end
            end

            -- Add template error categories.
            if cfg.templateErrorCategory then
                    local templateErrorCategory = cfg.templateErrorCategory
                    local templateCat, templateSort
                    if not self.name and not self.title.isSubpage then
                            templateCat = templateErrorCategory
                    elseif self.isTemplatePage then
                            local paramsToCheck = cfg.templateErrorParamsToCheck or {
                            local count = 0
                            for i, param in ipairs(paramsToCheck) do
                                    if not args[param] then
                                            count = count + 1
                                    end
                            end
                            if count > 0 then
                                    templateCat = templateErrorCategory
                                    templateSort = tostring(count)
                            end
                            if self.categoryNums and #self.categoryNums > 0 then
                                    templateCat = templateErrorCategory
                                    templateSort = 'C'
                            end
                    end
                    self:addCat(10, templateCat, templateSort)
            end
    end

    function MessageBox:setAllNamespaceCategories()
            -- Set categories for all namespaces.
            if self.invalidTypeError then
                    local allSort = (self.title.namespace == 0 and 'Main:' or '') ..
```

```
                    self:addCat('all', 'Wikipedia message box parameter needs fixing
            end
            if self.isSubstituted then
                    self:addCat('all', 'Pages with incorrectly substituted templates
            end
    end

    function MessageBox:setCategories()
            if self.title.namespace == 0 then
                    self:setMainspaceCategories()
            elseif self.title.namespace == 10 then
                    self:setTemplateCategories()
            end
            self:setAllNamespaceCategories()
    end

    function MessageBox:renderCategories()
            if not self.hasCategories then
                    -- No categories added, no need to pass them to Category handler
                    -- if it was invoked, it would return the empty string.
                    -- So we shortcut and return the empty string.
                    return ""
            end
            -- Convert category tables to strings and pass them through
            -- [[Module:Category handler]].
            return require('Module:Category handler')._main{
                    main = table.concat(self.categories[0] or {}),
                    template = table.concat(self.categories[10] or {}),
                    all = table.concat(self.categories.all or {}),
                    nocat = self.args.nocat,
                    page = self.args.page
            }
    end

    function MessageBox:export()
            local root = mw.html.create()

            -- Add the subst check error.
            if self.isSubstituted and self.name then
                    root:tag('b')
                            :addClass('error')
                            :wikitext(string.format(
                                    'Template <code>%s[[Template:%s|%s]]%s</code> has
                                    mw.text.nowiki('{{'), self.name, self.name, mw.te
                            ))
            end

            -- Create the box table.
            local boxTable = root:tag('table')
            boxTable:attr('id', self.id or nil)
            for i, class in ipairs(self.classes or {}) do
                    boxTable:addClass(class or nil)
            end
            boxTable
                    :cssText(self.style or nil)
                    :attr('role', 'presentation')

            if self.attrs then
                    boxTable:attr(self.attrs)
            end

            -- Add the left-hand image.
            local row = boxTable:tag('tr')
            if self.imageLeft then
```

```
                local imageLeftCell = row:tag('td'):addClass('mbox-image')
                if self.imageCellDiv then
                        -- If we are using a div, redefine imageLeftCell so that
                        -- is inside it. Divs use style="width: 52px;", which lim
                        -- image width to 52px. If any images in a div are wider
                        -- they may overlap with the text or cause other display
                        imageLeftCell = imageLeftCell:tag('div'):css('width', '52
                end
                imageLeftCell:wikitext(self.imageLeft or nil)
        elseif self.imageEmptyCell then
                -- Some message boxes define an empty cell if no image is specif
                -- some don't. The old template code in templates where empty cel
                -- specified gives the following hint: "No image. Cell with some
                -- or padding necessary for text cell to have 100% width."
                row:tag('td')
                        :addClass('mbox-empty-cell')
                        :cssText(self.imageEmptyCellStyle or nil)
        end

        -- Add the text.
        local textCell = row:tag('td'):addClass('mbox-text')
        if self.useCollapsibleTextFields then
                -- The message box uses advanced text parameters that allow thing
                -- collapsible. At the moment, only ambox uses this.
                textCell:cssText(self.textstyle or nil)
                local textCellDiv = textCell:tag('div')
                textCellDiv
                        :addClass('mbox-text-span')
                        :wikitext(self.issue or nil)
                if (self.talk or self.fix) then
                        textCellDiv:tag('span')
                                :addClass('hide-when-compact')
                                :wikitext(self.talk and (' ' .. self.talk) or nil
                                :wikitext(self.fix and (' ' .. self.fix) or nil)
                end
                textCellDiv:wikitext(self.date and (' ' .. self.date) or nil)
                if self.info and not self.isSmall then
                        textCellDiv
                                :tag('span')
                                :addClass('hide-when-compact')
                                :wikitext(self.info and (' ' .. self.info) or nil
                end
                if self.removalNotice then
                        textCellDiv:tag('span')
                                :addClass('hide-when-compact')
                                :tag('i')
                                        :wikitext(string.format(" (%s)", self.rem
                end
        else
                -- Default text formatting - anything goes.
                textCell
                        :cssText(self.textstyle or nil)
                        :wikitext(self.text or nil)
        end

        -- Add the right-hand image.
        if self.imageRight then
                local imageRightCell = row:tag('td'):addClass('mbox-imageright')
                if self.imageCellDiv then
                        -- If we are using a div, redefine imageRightCell so that
                        -- is inside it.
                        imageRightCell = imageRightCell:tag('div'):css('width', 
                end
                imageRightCell
```

```lua
                                :wikitext(self.imageRight or nil)
        end

        -- Add the below row.
        if self.below then
                boxTable:tag('tr')
                        :tag('td')
                                :attr('colspan', self.imageRight and '3' or '2')
                                :addClass('mbox-text')
                                :cssText(self.textstyle or nil)
                                :wikitext(self.below or nil)
        end

        -- Add error message for invalid type parameters.
        if self.invalidTypeError then
                root:tag('div')
                        :css('text-align', 'center')
                        :wikitext(string.format(
                                'This message box is using an invalid "type=%s" p
                                self.type or ''
                        ))
        end

        -- Add categories.
        root:wikitext(self:renderCategories() or nil)

        return tostring(root)
end

--------------------------------------------------------------------------------
-- Exports
--------------------------------------------------------------------------------

local p, mt = {}, {}

function p._exportClasses()
        -- For testing.
        return {
                MessageBox = MessageBox
        }
end

function p.main(boxType, args, cfgTables)
        local box = MessageBox.new(boxType, args, cfgTables or mw.loadData(CONFIG
        box:setParameters()
        box:setCategories()
        return box:export()
end

function mt.__index(t, k)
        return function (frame)
                if not getArgs then
                        getArgs = require('Module:Arguments').getArgs
                end
                return t.main(k, getArgs(frame, {trim = false, removeBlanks = fal
        end
end

return setmetatable(p, mt)
```

Modul:Message box/configuration

```lua
--------------------------------------------------------------------------------
--                          Message box configuration                        --
--                                                                           --
-- This module contains configuration data for [[Module:Message box]].       --
--------------------------------------------------------------------------------

return {
        ambox = {
                types = {
                        speedy = {
                                class = 'ambox-speedy',
                                image = 'Ambox warning pn.svg'
                        },
                        delete = {
                                class = 'ambox-delete',
                                image = 'Ambox warning pn.svg'
                        },
                        content = {
                                class = 'ambox-content',
                                image = 'Ambox important.svg'
                        },
                        style = {
                                class = 'ambox-style',
                                image = 'Edit-clear.svg'
                        },
                        move = {
                                class = 'ambox-move',
                                image = 'Merge-split-transwiki default.svg'
                        },
                        protection = {
                                class = 'ambox-protection',
                                image = 'Semi-protection-shackle-keyhole.svg'
                        },
                        notice = {
                                class = 'ambox-notice',
                                image = 'Information icon4.svg'
                        }
                },
                default                   = 'notice',
                allowBlankParams          = {'talk', 'sect', 'date', 'issue',
                allowSmall                = true,
                smallParam                = 'left',
                smallClass                = 'mbox-small-left',
                substCheck                = true,
                classes                   = {'metadata', 'ambox'},
                imageEmptyCell            = true,
                imageCheckBlank           = true,
                imageSmallSize            = '20x20px',
                imageCellDiv              = true,
                useCollapsibleTextFields  = true,
                imageRightNone            = true,
                sectionDefault            = 'article',
                allowMainspaceCategories  = true,
                templateCategory          = 'Article message templates',
                templateCategoryRequireName = true,
                templateErrorCategory     = 'Article message templates with mis
                templateErrorParamsToCheck = {'issue', 'fix', 'subst'},
                removalNotice             = '[[Help:Maintenance template remova
```

```
            },

        cmbox = {
                types = {
                        speedy = {
                                class = 'cmbox-speedy',
                                image = 'Ambox warning pn.svg'
                        },
                        delete = {
                                class = 'cmbox-delete',
                                image = 'Ambox warning pn.svg'
                        },
                        content = {
                                class = 'cmbox-content',
                                image = 'Ambox important.svg'
                        },
                        style = {
                                class = 'cmbox-style',
                                image = 'Edit-clear.svg'
                        },
                        move = {
                                class = 'cmbox-move',
                                image = 'Merge-split-transwiki default.svg'
                        },
                        protection = {
                                class = 'cmbox-protection',
                                image = 'Semi-protection-shackle-keyhole.svg'
                        },
                        notice = {
                                class = 'cmbox-notice',
                                image = 'Information icon4.svg'
                        }
                },
                default            = 'notice',
                showInvalidTypeError = true,
                classes            = {'cmbox'},
                imageEmptyCell     = true
        },

        fmbox = {
                types = {
                        warning = {
                                class = 'fmbox-warning',
                                image = 'Ambox warning pn.svg'
                        },
                        editnotice = {
                                class = 'fmbox-editnotice',
                                image = 'Information icon4.svg'
                        },
                        system = {
                                class = 'fmbox-system',
                                image = 'Information icon4.svg'
                        }
                },
                default            = 'system',
                showInvalidTypeError = true,
                classes            = {'fmbox'},
                imageEmptyCell     = false,
                imageRightNone     = false
        },

        imbox = {
                types = {
                        speedy = {
```

```lua
                                class = 'imbox-speedy',
                                image = 'Ambox warning pn.svg'
                        },
                        delete = {
                                class = 'imbox-delete',
                                image = 'Ambox warning pn.svg'
                        },
                        content = {
                                class = 'imbox-content',
                                image = 'Ambox important.svg'
                        },
                        style = {
                                class = 'imbox-style',
                                image = 'Edit-clear.svg'
                        },
                        move = {
                                class = 'imbox-move',
                                image = 'Merge-split-transwiki default.svg'
                        },
                        protection = {
                                class = 'imbox-protection',
                                image = 'Semi-protection-shackle-keyhole.svg'
                        },
                        license = {
                                class = 'imbox-license licensetpl',
                                image = 'Imbox license.png'  -- @todo We need an S
                        },
                        featured = {
                                class = 'imbox-featured',
                                image = 'Cscr-featured.svg'
                        },
                        notice = {
                                class = 'imbox-notice',
                                image = 'Information icon4.svg'
                        }
                },
                default             = 'notice',
                showInvalidTypeError = true,
                classes             = {'imbox'},
                imageEmptyCell      = true,
                below               = true,
                templateCategory    = 'File message boxes'
        },

        ombox = {
                types = {
                        speedy = {
                                class = 'ombox-speedy',
                                image = 'Ambox warning pn.svg'
                        },
                        delete = {
                                class = 'ombox-delete',
                                image = 'Ambox warning pn.svg'
                        },
                        content = {
                                class = 'ombox-content',
                                image = 'Ambox important.svg'
                        },
                        style = {
                                class = 'ombox-style',
                                image = 'Edit-clear.svg'
                        },
                        move = {
                                class = 'ombox-move',
```

```
                                image = 'Merge-split-transwiki default.svg'
                        },
                        protection = {
                                class = 'ombox-protection',
                                image = 'Semi-protection-shackle-keyhole.svg'
                        },
                        notice = {
                                class = 'ombox-notice',
                                image = 'Information icon4.svg'
                        }
                },
                default             = 'notice',
                showInvalidTypeError = true,
                classes             = {'ombox'},
                allowSmall          = true,
                imageEmptyCell      = true,
                imageRightNone      = true
        },

        tmbox = {
                types = {
                        speedy = {
                                class = 'tmbox-speedy',
                                image = 'Ambox warning pn.svg'
                        },
                        delete = {
                                class = 'tmbox-delete',
                                image = 'Ambox warning pn.svg'
                        },
                        content = {
                                class = 'tmbox-content',
                                image = 'Ambox important.svg'
                        },
                        style = {
                                class = 'tmbox-style',
                                image = 'Edit-clear.svg'
                        },
                        move = {
                                class = 'tmbox-move',
                                image = 'Merge-split-transwiki default.svg'
                        },
                        protection = {
                                class = 'tmbox-protection',
                                image = 'Semi-protection-shackle-keyhole.svg'
                        },
                        notice = {
                                class = 'tmbox-notice',
                                image = 'Information icon4.svg'
                        }
                },
                default             = 'notice',
                showInvalidTypeError = true,
                classes             = {'tmbox'},
                allowSmall          = true,
                imageRightNone      = true,
                imageEmptyCell      = true,
                imageEmptyCellStyle = true,
                templateCategory    = 'Talk message boxes'
        }
}
```