Modul:Documentation

Vorlage:Lua

This module displays a blue box containing documentation for templates, Lua modules, or other pages. The Vorlage:Tl template invokes it.

Normal usage

For most uses, you should use the Vorlage:Tl template; please see that template's page for its usage instructions and parameters.

Use in other modules

To use this module from another Lua module, first load it with `require`:

```
local documentation = require('Module:Documentation').main
```

Then you can simply call it using a table of arguments.

```
documentation{content = 'Some documentation', ['link box'] = 'My custom link box'
```

Please refer to the template documentation for usage instructions and a list of parameters.

Porting to other wikis

The module has a configuration file at Module:Documentation/config which is intended to allow easy translation and porting to other wikis. Please see the code comments in the config page for instructions. If you have any questions, or you need a feature which is not currently implemented, please leave a message at Template talk:Documentation to get the attention of a developer.

The messages that need to be customized to display a documentation template/module at the top of module pages are MediaWiki:Scribunto-doc-page-show and MediaWiki:Scribunto-doc-page-does-not-exist.

```
-- This module implements {{documentation}}.

-- Get required modules.
local getArgs = require('Module:Arguments').getArgs
local messageBox = require('Module:Message box')

-- Get the config table.
local cfg = mw.loadData('Module:Documentation/config')
local i18n = mw.loadData('Module:Documentation/i18n')
local p = {}

-- Often-used functions.
```

```
local ugsub = mw.ustring.gsub

--------------------------------------------------------------------------------
-- Helper functions
--
-- These are defined as local functions, but are made available in the p
-- table for testing purposes.
--------------------------------------------------------------------------------

local function message(cfgKey, valArray, expectType)
        --[[
        -- Gets a message from the cfg table and formats it if appropriate.
        -- The function raises an error if the value from the cfg table is not
        -- of the type expectType. The default type for expectType is 'string'.
        -- If the table valArray is present, strings such as $1, $2 etc. in the
        -- message are substituted with values from the table keys [1], [2] etc.
        -- For example, if the message "foo-message" had the value 'Foo $2 bar $1
        -- message('foo-message', {'baz', 'qux'}) would return "Foo qux bar baz."
        --]]
        local msg = cfg[cfgKey]
        expectType = expectType or 'string'
        if type(msg) ~= expectType then
                error(require('Module:TNT').format('I18n/Documentation', 'cfg-er
        end
        if not valArray then
                return msg
        end

        local function getMessageVal(match)
                match = tonumber(match)
                return valArray[match] or error(require('Module:TNT').format('I1
        end

        local ret = ugsub(msg, '$([1-9][0-9]*)', getMessageVal)
        return ret
end

p.message = message

local function makeWikilink(page, display)
        if display then
                return mw.ustring.format('[[%s|%s]]', page, display)
        else
                return mw.ustring.format('[[%s]]', page)
        end
end

p.makeWikilink = makeWikilink

local function makeCategoryLink(cat, sort)
        local catns = mw.site.namespaces[14].name
        return makeWikilink(catns .. ':' .. cat, sort)
end

p.makeCategoryLink = makeCategoryLink

local function makeUrlLink(url, display)
        return mw.ustring.format('[%s %s]', url, display)
end

p.makeUrlLink = makeUrlLink

local function makeToolbar(...)
        local ret = {}
```

```
	local lim = select('#', ...)
	if lim < 1 then
		return nil
	end
	for i = 1, lim do
		ret[#ret + 1] = select(i, ...)
	end
	return '<small style="font-style: normal;">(' .. table.concat(ret, ' &#12
end

p.makeToolbar = makeToolbar

--------------------------------------------------------------------------------
-- Argument processing
--------------------------------------------------------------------------------

local function makeInvokeFunc(funcName)
	return function (frame)
		local args = getArgs(frame, {
			valueFunc = function (key, value)
				if type(value) == 'string' then
					value = value:match('^%s*(.-)%s*$') -- Re
					if key == 'heading' or value ~= '' then
						return value
					else
						return nil
					end
				else
					return value
				end
			end
		})
		return p[funcName](args)
	end
end

--------------------------------------------------------------------------------
-- Load TemplateStyles
--------------------------------------------------------------------------------

p.main = function(frame)
	local parent = frame.getParent(frame)
	local output = p._main(parent.args)
	return frame:extensionTag{ name='templatestyles', args = { src= message('
end

--------------------------------------------------------------------------------
-- Main function
--------------------------------------------------------------------------------

function p._main(args)
	--[[
	-- This function defines logic flow for the module.
	-- @args - table of arguments passed by the user
	--
	-- Messages:
	-- 'main-div-id' --> 'template-documentation'
	-- 'main-div-classes' --> 'template-documentation iezoomfix'
	--]]
	local env = p.getEnvironment(args)
	local root = mw.html.create()
	root
		:wikitext(p.protectionTemplate(env))
		:wikitext(p.sandboxNotice(args, env))
```

```
                -- This div tag is from {{documentation/start box}}, but moving
                -- so that we don't have to worry about unclosed tags.
                :tag('div')
                        :attr('id', message('main-div-id'))
                        :addClass(message('main-div-class'))
                        :wikitext(p._startBox(args, env))
                        :wikitext(p._content(args, env))
                        :done()
                :wikitext(p._endBox(args, env))
                :wikitext(p.addTrackingCategories(env))
        return tostring(root)
end

--------------------------------------------------------------------------------
-- Environment settings
--------------------------------------------------------------------------------

function p.getEnvironment(args)
        --[[
        -- Returns a table with information about the environment, including titl
        -- path-related data.
        -- @args - table of arguments passed by the user
        --
        -- Title objects include:
        -- env.title - the page we are making documentation for (usually the curr
        -- env.templateTitle - the template (or module, file, etc.)
        -- env.docTitle - the /doc subpage.
        -- env.sandboxTitle - the /sandbox subpage.
        -- env.testcasesTitle - the /testcases subpage.
        -- env.printTitle - the print version of the template, located at the /Pr
        --
        -- Data includes:
        -- env.protectionLevels - the protection levels table of the title object
        -- env.subjectSpace - the number of the title's subject namespace.
        -- env.docSpace - the number of the namespace the title puts its document
        -- env.docpageBase - the text of the base page of the /doc, /sandbox and
        -- env.compareUrl - URL of the Special:ComparePages page comparing the sa
        --
        -- All table lookups are passed through pcall so that errors are caught.
        -- returned will be nil.
        --]]

        local env, envFuncs = {}, {}

        -- Set up the metatable. If triggered we call the corresponding function
        -- returned by that function is memoized in the env table so that we don'
        -- more than once. (Nils won't be memoized.)
        setmetatable(env, {
                __index = function (t, key)
                        local envFunc = envFuncs[key]
                        if envFunc then
                                local success, val = pcall(envFunc)
                                if success then
                                        env[key] = val -- Memoise the value.
                                        return val
                                end
                        end
                        return nil
                end
        })

        function envFuncs.title()
                -- The title object for the current page, or a test page passed v
                local title
```

```
		local titleArg = args.page
		if titleArg then
			title = mw.title.new(titleArg)
		else
			title = mw.title.getCurrentTitle()
		end
		return title
	end

	function envFuncs.templateTitle()
		--[[
		-- The template (or module, etc.) title object.
		-- Messages:
		-- 'sandbox-subpage' --> 'sandbox'
		-- 'testcases-subpage' --> 'testcases'
		--]]
		local subjectSpace = env.subjectSpace
		local title = env.title
		local subpage = title.subpageText
		if subpage == message('sandbox-subpage') or subpage == message('t
			return mw.title.makeTitle(subjectSpace, title.baseText)
		else
			return mw.title.makeTitle(subjectSpace, title.text)
		end
	end

	function envFuncs.docTitle()
		--[[
		-- Title object of the /doc subpage.
		-- Messages:
		-- 'doc-subpage' --> 'doc'
		--]]
		local title = env.title
		local docname = args[1] -- User-specified doc page.
		local docpage
		if docname then
			docpage = docname
		else
			docpage = env.docpageBase .. '/' .. message('doc-subpage'
		end
		return mw.title.new(docpage)
	end

	function envFuncs.sandboxTitle()
		--[[
		-- Title object for the /sandbox subpage.
		-- Messages:
		-- 'sandbox-subpage' --> 'sandbox'
		--]]
		return mw.title.new(env.docpageBase .. '/' .. message('sandbox-su
	end

	function envFuncs.testcasesTitle()
		--[[
		-- Title object for the /testcases subpage.
		-- Messages:
		-- 'testcases-subpage' --> 'testcases'
		--]]
		return mw.title.new(env.docpageBase .. '/' .. message('testcases-
	end

	function envFuncs.printTitle()
		--[[
		-- Title object for the /Print subpage.
```

```
                -- Messages:
                -- 'print-subpage' --> 'Print'
                --]]
                return env.templateTitle:subPageTitle(message('print-subpage'))
        end

        function envFuncs.protectionLevels()
                -- The protection levels table of the title object.
                return env.title.protectionLevels
        end

        function envFuncs.subjectSpace()
                -- The subject namespace number.
                return mw.site.namespaces[env.title.namespace].subject.id
        end

        function envFuncs.docSpace()
                -- The documentation namespace number. For most namespaces this i
                -- subject namespace. However, pages in the Article, File, MediaW
                -- namespaces must have their /doc, /sandbox and /testcases pages
                local subjectSpace = env.subjectSpace
                if subjectSpace == 0 or subjectSpace == 6 or subjectSpace == 8 or
                        return subjectSpace + 1
                else
                        return subjectSpace
                end
        end

        function envFuncs.docpageBase()
                -- The base page of the /doc, /sandbox, and /testcases subpages.
                -- For some namespaces this is the talk page, rather than the tem
                local templateTitle = env.templateTitle
                local docSpace = env.docSpace
                local docSpaceText = mw.site.namespaces[docSpace].name
                -- Assemble the link. docSpace is never the main namespace, so we
                return docSpaceText .. ':' .. templateTitle.text
        end

        function envFuncs.compareUrl()
                -- Diff link between the sandbox and the main template using [[Sp
                local templateTitle = env.templateTitle
                local sandboxTitle = env.sandboxTitle
                if templateTitle.exists and sandboxTitle.exists then
                        local compareUrl = mw.uri.fullUrl(
                                'Special:ComparePages',
                                {page1 = templateTitle.prefixedText, page2 = sand
                        )
                        return tostring(compareUrl)
                else
                        return nil
                end
        end

        return env
end

--------------------------------------------------------------------------------
-- Auxiliary templates
--------------------------------------------------------------------------------

function p.sandboxNotice(args, env)
        --[=[
        -- Generates a sandbox notice for display above sandbox pages.
        -- @args - a table of arguments passed by the user
```

```
	-- @env - environment table containing title objects, etc., generated wi
	--
	-- Messages:
	-- 'sandbox-notice-image' --> '[[Image:Sandbox.svg|50px|alt=|link=]]'
	-- 'sandbox-notice-blurb' --> 'This is the $1 for $2.'
	-- 'sandbox-notice-diff-blurb' --> 'This is the $1 for $2 ($3).'
	-- 'sandbox-notice-pagetype-template' --> '[[w:Wikipedia:Template test ca
	-- 'sandbox-notice-pagetype-module' --> '[[w:Wikipedia:Template test case
	-- 'sandbox-notice-pagetype-other' --> 'sandbox page'
	-- 'sandbox-notice-compare-link-display' --> 'diff'
	-- 'sandbox-notice-testcases-blurb' --> 'See also the companion subpage f
	-- 'sandbox-notice-testcases-link-display' --> 'test cases'
	-- 'sandbox-category' --> 'Template sandboxes'
	--]=]
	local title = env.title
	local sandboxTitle = env.sandboxTitle
	local templateTitle = env.templateTitle
	local subjectSpace = env.subjectSpace
	if not (subjectSpace and title and sandboxTitle and templateTitle and mw.
		return nil
	end
	-- Build the table of arguments to pass to {{ombox}}. We need just two fi
	local omargs = {}
	omargs.image = message('sandbox-notice-image')
	-- Get the text. We start with the opening blurb, which is something like
	-- "This is the template sandbox for [[Template:Foo]] (diff)."
	local text = ''
	local frame = mw.getCurrentFrame()
	local isPreviewing = frame:preprocess('{{REVISIONID}}') == '' -- True if
	local pagetype
	if subjectSpace == 10 then
		pagetype = message('sandbox-notice-pagetype-template')
	elseif subjectSpace == 828 then
		pagetype = message('sandbox-notice-pagetype-module')
	else
		pagetype = message('sandbox-notice-pagetype-other')
	end
	local templateLink = makeWikilink(templateTitle.prefixedText)
	local compareUrl = env.compareUrl
	if isPreviewing or not compareUrl then
		text = text .. message('sandbox-notice-blurb', {pagetype, templat
	else
		local compareDisplay = message('sandbox-notice-compare-link-displ
		local compareLink = makeUrlLink(compareUrl, compareDisplay)
		text = text .. message('sandbox-notice-diff-blurb', {pagetype, te
	end
	-- Get the test cases page blurb if the page exists. This is something li
	-- "See also the companion subpage for [[Template:Foo/testcases|test case
	local testcasesTitle = env.testcasesTitle
	if testcasesTitle and testcasesTitle.exists then
		if testcasesTitle.contentModel == "Scribunto" then
			local testcasesLinkDisplay = message('sandbox-notice-test
			local testcasesRunLinkDisplay = message('sandbox-notice-t
			local testcasesLink = makeWikilink(testcasesTitle.prefixe
			local testcasesRunLink = makeWikilink(testcasesTitle.talk
			text = text .. '<br />' .. message('sandbox-notice-testca
		else
			local testcasesLinkDisplay = message('sandbox-notice-test
			local testcasesLink = makeWikilink(testcasesTitle.prefixe
			text = text .. '<br />' .. message('sandbox-notice-testca
		end
	end
	-- Add the sandbox to the sandbox category.
	text = text .. makeCategoryLink(message('sandbox-category'))
```

```
        omargs.text = text
        omargs.class = message('sandbox-class')
        local ret = '<div style="clear: both;"></div>'
        ret = ret .. messageBox.main('ombox', omargs)
        return ret
end

function p.protectionTemplate(env)
        -- Generates the padlock icon in the top right.
        -- @env - environment table containing title objects, etc., generated wit
        -- Messages:
        -- 'protection-template' --> 'pp-template'
        -- 'protection-template-args' --> {docusage = 'yes'}
        local title = env.title
        local protectionLevels
        local protectionTemplate = message('protection-template')
        local namespace = title.namespace
        if not (protectionTemplate and (namespace == 10 or namespace == 828)) the
                -- Don't display the protection template if we are not in the tem
                return nil
        end
        protectionLevels = env.protectionLevels
        if not protectionLevels then
                return nil
        end
        local editLevels = protectionLevels.edit
        local moveLevels = protectionLevels.move
        if moveLevels and moveLevels[1] == 'sysop' or editLevels and editLevels[1
                -- The page is full-move protected, or full, template, or semi-pr
                local frame = mw.getCurrentFrame()
                return frame:expandTemplate{title = protectionTemplate, args = me
        else
                return nil
        end
end

--------------------------------------------------------------------------------
-- Start box
--------------------------------------------------------------------------------

p.startBox = makeInvokeFunc('_startBox')

function p._startBox(args, env)
        --[[
        -- This function generates the start box.
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated wit
        --
        -- The actual work is done by p.makeStartBoxLinksData and p.renderStartBo
        -- the [view] [edit] [history] [purge] links, and by p.makeStartBoxData a
        -- which generate the box HTML.
        --]]
        env = env or p.getEnvironment(args)
        local links
        local content = args.content
        if not content then
                -- No need to include the links if the documentation is on the te
                local linksData = p.makeStartBoxLinksData(args, env)
                if linksData then
                        links = p.renderStartBoxLinks(linksData)
                end
        end
        -- Generate the start box html.
        local data = p.makeStartBoxData(args, env, links)
```

```
	if data then
		return p.renderStartBox(data)
	else
		-- User specified no heading.
		return nil
	end
end

function p.makeStartBoxLinksData(args, env)
	--[[
	-- Does initial processing of data to make the [view] [edit] [history] [p
	-- @args - a table of arguments passed by the user
	-- @env - environment table containing title objects, etc., generated wit
	--
	-- Messages:
	-- 'view-link-display' --> 'view'
	-- 'edit-link-display' --> 'edit'
	-- 'history-link-display' --> 'history'
	-- 'purge-link-display' --> 'purge'
	-- 'file-docpage-preload' --> 'Template:Documentation/preload-filespace'
	-- 'module-preload' --> 'Template:Documentation/preload-module-doc'
	-- 'docpage-preload' --> 'Template:Documentation/preload'
	-- 'create-link-display' --> 'create'
	--]]
	local subjectSpace = env.subjectSpace
	local title = env.title
	local docTitle = env.docTitle
	if not title or not docTitle then
		return nil
	end
	if docTitle.isRedirect then
		docTitle = docTitle.redirectTarget
	end

	local data = {}
	data.title = title
	data.docTitle = docTitle
	-- View, display, edit, and purge links if /doc exists.
	data.viewLinkDisplay = i18n['view-link-display']
	data.editLinkDisplay = i18n['edit-link-display']
	data.historyLinkDisplay = i18n['history-link-display']
	data.purgeLinkDisplay = i18n['purge-link-display']
	-- Create link if /doc doesn't exist.
	local preload = args.preload
	if not preload then
		if subjectSpace == 6 then -- File namespace
			preload = message('file-docpage-preload')
		elseif subjectSpace == 828 then -- Module namespace
			preload = message('module-preload')
		else
			preload = message('docpage-preload')
		end
	end
	data.preload = preload
	data.createLinkDisplay = i18n['create-link-display']
	return data
end

function p.renderStartBoxLinks(data)
	--[[
	-- Generates the [view][edit][history][purge] or [create] links from the
	-- @data - a table of data generated by p.makeStartBoxLinksData
	--]]
```

```
	local function escapeBrackets(s)
		-- Escapes square brackets with HTML entities.
		s = s:gsub('%[', '&#91;') -- Replace square brackets with HTML er
		s = s:gsub('%]', '&#93;')
		return s
	end

	local ret
	local docTitle = data.docTitle
	local title = data.title
	if docTitle.exists then
		local viewLink = makeWikilink(docTitle.prefixedText, data.viewLi
		local editLink = makeUrlLink(docTitle:fullUrl{action = 'edit'}, 
		local historyLink = makeUrlLink(docTitle:fullUrl{action = 'histo
		local purgeLink = makeUrlLink(title:fullUrl{action = 'purge'}, da
		ret = '[%s] [%s] [%s] [%s]'
		ret = escapeBrackets(ret)
		ret = mw.ustring.format(ret, viewLink, editLink, historyLink, pu
	else
		local createLink = makeUrlLink(docTitle:fullUrl{action = 'edit',
		ret = '[%s]'
		ret = escapeBrackets(ret)
		ret = mw.ustring.format(ret, createLink)
	end
	return ret
end

function p.makeStartBoxData(args, env, links)
	--[=[
	-- Does initial processing of data to pass to the start-box render functi
	-- @args - a table of arguments passed by the user
	-- @env - environment table containing title objects, etc., generated wit
	-- @links - a string containing the [view][edit][history][purge] links -
	--
	-- Messages:
	-- 'documentation-icon-wikitext' --> '[[File:Test Template Info-Icon - Ve
	-- 'template-namespace-heading' --> 'Template documentation'
	-- 'module-namespace-heading' --> 'Module documentation'
	-- 'file-namespace-heading' --> 'Summary'
	-- 'other-namespaces-heading' --> 'Documentation'
	-- 'start-box-linkclasses' --> 'mw-editsection-like plainlinks'
	-- 'start-box-link-id' --> 'doc_editlinks'
	-- 'testcases-create-link-display' --> 'create'
	--]=]
	local subjectSpace = env.subjectSpace
	if not subjectSpace then
		-- Default to an "other namespaces" namespace, so that we get at
		-- if an error occurs.
		subjectSpace = 2
	end
	local data = {}

	-- Heading
	local heading = args.heading -- Blank values are not removed.
	if heading == '' then
		-- Don't display the start box if the heading arg is defined but
		return nil
	end
	if heading then
		data.heading = heading
	elseif subjectSpace == 10 then -- Template namespace
		data.heading = i18n['template-namespace-heading']
	elseif subjectSpace == 828 then -- Module namespace
		data.heading = i18n['module-namespace-heading']
```

```
	elseif subjectSpace == 6 then -- File namespace
		data.heading = i18n['file-namespace-heading']
	else
		data.heading = i18n['other-namespaces-heading']
	end

	-- Data for the [view][edit][history][purge] or [create] links.
	if links then
		data.linksClass = message('start-box-linkclasses')
		data.linksId = message('start-box-link-id')
		data.links = links
	end

	return data
end

function p.renderStartBox(data)
	-- Renders the start box html.
	-- @data - a table of data generated by p.makeStartBoxData.
	local sbox = mw.html.create('div')
	sbox
		:addClass(message('header-div-class'))
		:tag('div')
			:addClass(message('heading-div-class'))
			:wikitext(data.heading)
	local links = data.links
	if links then
		sbox
			:tag('div')
				:addClass(data.linksClass)
				:attr('id', data.linksId)
				:wikitext(links)
	end
	return tostring(sbox)
end

--------------------------------------------------------------------------------
-- Documentation content
--------------------------------------------------------------------------------

p.content = makeInvokeFunc('_content')

function p._content(args, env)
	-- Displays the documentation contents
	-- @args - a table of arguments passed by the user
	-- @env - environment table containing title objects, etc., generated wi
	env = env or p.getEnvironment(args)
	local docTitle = env.docTitle
	local content = args.content
	if not content and docTitle and docTitle.exists then
		content = args._content or mw.getCurrentFrame():expandTemplate{ti
	end
	-- The line breaks below are necessary so that "=== Headings ===" at the
	-- of docs are interpreted correctly.
	local cbox = mw.html.create('div')
	cbox
		:addClass(message('content-div-class'))
		:wikitext('\n' .. (content or '') .. '\n')
	return tostring(cbox)
end

p.contentTitle = makeInvokeFunc('_contentTitle')

function p._contentTitle(args, env)
```

```
	env = env or p.getEnvironment(args)
	local docTitle = env.docTitle
	if not args.content and docTitle and docTitle.exists then
		return docTitle.prefixedText
	else
		return ''
	end
end

--------------------------------------------------------------------------------
-- End box
--------------------------------------------------------------------------------

p.endBox = makeInvokeFunc('_endBox')

function p._endBox(args, env)
	--[=[
	-- This function generates the end box (also known as the link box).
	-- @args - a table of arguments passed by the user
	-- @env - environment table containing title objects, etc., generated wi
	--]=]

	-- Get environment data.
	env = env or p.getEnvironment(args)
	local subjectSpace = env.subjectSpace
	local docTitle = env.docTitle
	if not subjectSpace or not docTitle then
		return nil
	end

	-- Check whether we should output the end box at all. Add the end
	-- box by default if the documentation exists or if we are in the
	-- user, module or template namespaces.
	local linkBox = args['link box']
	if linkBox == 'off'
		or not (
			docTitle.exists
			or subjectSpace == 2
			or subjectSpace == 828
			or subjectSpace == 10
		)
	then
		return nil
	end

	-- Assemble the footer text field.
	local text = ''
	if linkBox then
		text = text .. linkBox
	else
		text = text .. (p.makeDocPageBlurb(args, env) or '') -- "This doc
		if subjectSpace == 2 or subjectSpace == 10 or subjectSpace == 828
			-- We are in the user, template or module namespaces.
			-- Add sandbox and testcases links.
			-- "Editors can experiment in this template's sandbox and
			text = text .. (p.makeExperimentBlurb(args, env) or '')
			text = text .. '<br />'
			if not args.content and not args[1] then
				-- "Please add categories to the /doc subpage."
				-- Don't show this message with inline docs or wi
				-- as then it is unclear where to add the categor
				text = text .. (p.makeCategoriesBlurb(args, env)
			end
			text = text .. ' ' .. (p.makeSubpagesBlurb(args, env) or
```

```
                        local printBlurb = p.makePrintBlurb(args, env) -- Two-lin
                        if printBlurb then
                                text = text .. '<br />' .. printBlurb
                        end
                end
        end

        local ebox = mw.html.create('div')
        ebox
                :addClass(message('footer-div-class'))
                :wikitext(text)
        return tostring(ebox)
end

function p.makeDocPageBlurb(args, env)
        --[=[
        -- Makes the blurb "This documentation is transcluded from [[Template:Foo
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated wit
        --
        -- Messages:
        -- 'edit-link-display' --> 'edit'
        -- 'history-link-display' --> 'history'
        -- 'transcluded-from-blurb' -->
        -- 'The above [[w:Wikipedia:Template documentation|documentation]]
        -- is [[w:Wikipedia:Transclusion|transcluded]] from $1.'
        -- 'module-preload' --> 'Template:Documentation/preload-module-doc'
        -- 'create-link-display' --> 'create'
        -- 'create-module-doc-blurb' -->
        -- 'You might want to $1 a documentation page for this [[w:Wikipedia:Lua|
        --]=]
        local docTitle = env.docTitle
        if not docTitle or args.content then
                return nil
        end
        local ret
        if docTitle.exists then
                -- /doc exists; link to it.
                local docLink = makeWikilink(docTitle.prefixedText)
                local editUrl = docTitle:fullUrl{action = 'edit'}
                local editDisplay = i18n['edit-link-display']
                local editLink = makeUrlLink(editUrl, editDisplay)
                local historyUrl = docTitle:fullUrl{action = 'history'}
                local historyDisplay = i18n['history-link-display']
                local historyLink = makeUrlLink(historyUrl, historyDisplay)
                ret = message('transcluded-from-blurb', {docLink})
                        .. ' '
                        .. makeToolbar(editLink, historyLink)
                        .. '<br />'
        elseif env.subjectSpace == 828 then
                -- /doc does not exist; ask to create it.
                local createUrl = docTitle:fullUrl{action = 'edit', preload = mes
                local createDisplay = i18n['create-link-display']
                local createLink = makeUrlLink(createUrl, createDisplay)
                ret = message('create-module-doc-blurb', {createLink})
                        .. '<br />'
        end
        return ret
end

function p.makeExperimentBlurb(args, env)
        --[[
        -- Renders the text "Editors can experiment in this template's sandbox (e
        -- @args - a table of arguments passed by the user
```

```
	-- @env - environment table containing title objects, etc., generated wi
	--
	-- Messages:
	-- 'sandbox-link-display' --> 'sandbox'
	-- 'sandbox-edit-link-display' --> 'edit'
	-- 'compare-link-display' --> 'diff'
	-- 'module-sandbox-preload' --> 'Template:Documentation/preload-module-sa
	-- 'template-sandbox-preload' --> 'Template:Documentation/preload-sandbox
	-- 'sandbox-create-link-display' --> 'create'
	-- 'mirror-edit-summary' --> 'Create sandbox version of $1'
	-- 'mirror-link-display' --> 'mirror'
	-- 'mirror-link-preload' --> 'Template:Documentation/mirror'
	-- 'sandbox-link-display' --> 'sandbox'
	-- 'testcases-link-display' --> 'testcases'
	-- 'testcases-edit-link-display'--> 'edit'
	-- 'template-sandbox-preload' --> 'Template:Documentation/preload-sandbox
	-- 'testcases-create-link-display' --> 'create'
	-- 'testcases-link-display' --> 'testcases'
	-- 'testcases-edit-link-display' --> 'edit'
	-- 'module-testcases-preload' --> 'Template:Documentation/preload-module-
	-- 'template-testcases-preload' --> 'Template:Documentation/preload-teste
	-- 'experiment-blurb-module' --> 'Editors can experiment in this module's
	-- 'experiment-blurb-template' --> 'Editors can experiment in this templa
	--]]
	local subjectSpace = env.subjectSpace
	local templateTitle = env.templateTitle
	local sandboxTitle = env.sandboxTitle
	local testcasesTitle = env.testcasesTitle
	local templatePage = templateTitle.prefixedText
	if not subjectSpace or not templateTitle or not sandboxTitle or not teste
		return nil
	end
	-- Make links.
	local sandboxLinks, testcasesLinks
	if sandboxTitle.exists then
		local sandboxPage = sandboxTitle.prefixedText
		local sandboxDisplay = message('sandbox-link-display')
		local sandboxLink = makeWikilink(sandboxPage, sandboxDisplay)
		local sandboxEditUrl = sandboxTitle:fullUrl{action = 'edit'}
		local sandboxEditDisplay = message('sandbox-edit-link-display')
		local sandboxEditLink = makeUrlLink(sandboxEditUrl, sandboxEditDi
		local compareUrl = env.compareUrl
		local compareLink
		if compareUrl then
			local compareDisplay = message('compare-link-display')
			compareLink = makeUrlLink(compareUrl, compareDisplay)
		end
		sandboxLinks = sandboxLink .. ' ' .. makeToolbar(sandboxEditLink,
	else
		local sandboxPreload
		if subjectSpace == 828 then
			sandboxPreload = message('module-sandbox-preload')
		else
			sandboxPreload = message('template-sandbox-preload')
		end
		local sandboxCreateUrl = sandboxTitle:fullUrl{action = 'edit', pr
		local sandboxCreateDisplay = message('sandbox-create-link-display
		local sandboxCreateLink = makeUrlLink(sandboxCreateUrl, sandboxCr
		local mirrorSummary = message('mirror-edit-summary', {makeWikilin
		local mirrorPreload = message('mirror-link-preload')
		local mirrorUrl = sandboxTitle:fullUrl{action = 'edit', preload =
		local mirrorDisplay = message('mirror-link-display')
		local mirrorLink = makeUrlLink(mirrorUrl, mirrorDisplay)
		sandboxLinks = message('sandbox-link-display') .. ' ' .. makeTool
```

```
        end
        if testcasesTitle.exists then
                local testcasesPage = testcasesTitle.prefixedText
                local testcasesDisplay = message('testcases-link-display')
                local testcasesLink = makeWikilink(testcasesPage, testcasesDispla
                local testcasesEditUrl = testcasesTitle:fullUrl{action = 'edit'}
                local testcasesEditDisplay = message('testcases-edit-link-display
                local testcasesEditLink = makeUrlLink(testcasesEditUrl, testcases
                testcasesLinks = testcasesLink .. ' ' .. makeToolbar(testcasesEdi
        else
                local testcasesPreload
                if subjectSpace == 828 then
                        testcasesPreload = message('module-testcases-preload')
                else
                        testcasesPreload = message('template-testcases-preload')
                end
                local testcasesCreateUrl = testcasesTitle:fullUrl{action = 'edit'
                local testcasesCreateDisplay = message('testcases-create-link-dis
                local testcasesCreateLink = makeUrlLink(testcasesCreateUrl, testc
                testcasesLinks = message('testcases-link-display') .. ' ' .. make
        end
        local messageName
        if subjectSpace == 828 then
                messageName = 'experiment-blurb-module'
        else
                messageName = 'experiment-blurb-template'
        end
        return message(messageName, {sandboxLinks, testcasesLinks})
end

function p.makeCategoriesBlurb(args, env)
        --[[
        -- Generates the text "Please add categories to the /doc subpage."
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated wit
        -- Messages:
        -- 'doc-link-display' --> '/doc'
        -- 'add-categories-blurb' --> 'Please add categories to the $1 subpage.'
        --]]
        local docTitle = env.docTitle
        if not docTitle then
                return nil
        end
        local docPathLink = makeWikilink(docTitle.prefixedText, message('doc-link
        return message('add-categories-blurb', {docPathLink})
end

function p.makeSubpagesBlurb(args, env)
        --[[
        -- Generates the "Subpages of this template" link.
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated wit

        -- Messages:
        -- 'template-pagetype' --> 'template'
        -- 'module-pagetype' --> 'module'
        -- 'default-pagetype' --> 'page'
        -- 'subpages-link-display' --> 'Subpages of this $1'
        --]]
        local subjectSpace = env.subjectSpace
        local templateTitle = env.templateTitle
        if not subjectSpace or not templateTitle then
                return nil
        end
```

```
	local pagetype
	if subjectSpace == 10 then
		pagetype = message('template-pagetype')
	elseif subjectSpace == 828 then
		pagetype = message('module-pagetype')
	else
		pagetype = message('default-pagetype')
	end
	local subpagesLink = makeWikilink(
		'Special:PrefixIndex/' .. templateTitle.prefixedText .. '/',
		message('subpages-link-display', {pagetype})
	)
	return message('subpages-blurb', {subpagesLink})
end

function p.makePrintBlurb(args, env)
	--[=[
	-- Generates the blurb displayed when there is a print version of the tem
	-- @args - a table of arguments passed by the user
	-- @env - environment table containing title objects, etc., generated wit
	--
	-- Messages:
	-- 'print-link-display' --> '/Print'
	-- 'print-blurb' --> 'A [[Help:Books/for experts#Improving the book layou
	--                 .. ' of this template exists at $1.'
	--                 .. ' If you make a change to this template, please upda
	-- 'display-print-category' --> true
	-- 'print-category' --> 'Templates with print versions'
	--]=]
	local printTitle = env.printTitle
	if not printTitle then
		return nil
	end
	local ret
	if printTitle.exists then
		local printLink = makeWikilink(printTitle.prefixedText, message('
		ret = message('print-blurb', {printLink})
		local displayPrintCategory = message('display-print-category', ni
		if displayPrintCategory then
			ret = ret .. makeCategoryLink(message('print-category'))
		end
	end
	return ret
end

--------------------------------------------------------------------------------
-- Tracking categories
--------------------------------------------------------------------------------

function p.addTrackingCategories(env)
	--[[
	-- Check if {{documentation}} is transcluded on a /doc or /testcases page
	-- @env - environment table containing title objects, etc., generated wit

	-- Messages:
	-- 'display-strange-usage-category' --> true
	-- 'doc-subpage' --> 'doc'
	-- 'testcases-subpage' --> 'testcases'
	-- 'strange-usage-category' --> 'Wikipedia pages with strange ((documenta
	--
	-- /testcases pages in the module namespace are not categorised, as they
	-- {{documentation}} transcluded automatically.
	--]]
	local title = env.title
```

```
	local subjectSpace = env.subjectSpace
	if not title or not subjectSpace then
		return nil
	end
	local subpage = title.subpageText
	local ret = ''
	if message('display-strange-usage-category', nil, 'boolean')
		and (
			subpage == message('doc-subpage')
			or subjectSpace ~= 828 and subpage == message('testcases-
		)
	then
		ret = ret .. makeCategoryLink(message('strange-usage-category'))
	end
	return ret
end

return p
```