Inhaltsverzeichnis

Modul:Uses TemplateStyles

Vorlage:Lua

Implements Vorlage:Tlx.

[[Category:Global modules{{#translation:}}]]

```
-- This module implements the {{Uses TemplateStyles}} template.
local yesno = require('Module:Yesno')
local mList = require('Module:List')
local mTableTools = require('Module:TableTools')
local mMessageBox = require('Module:Message box')
local TNT = require('Module:TNT')

local p = {}

local function format(msg, ...)
	return TNT.format('I18n/Uses TemplateStyles', msg, ...)
end

local function getConfig()
	return mw.loadData('Module:Uses TemplateStyles/config')
end

function p.main(frame)
	local origArgs = frame:getParent().args
	local args = {}
	for k, v in pairs(origArgs) do
		v = v:match('^%s*(.-)%s*$')
		if v ~= '' then
			args[k] = v
		end
	end
	return p._main(args)
end

function p._main(args, cfg)
	local tStyles = mTableTools.compressSparseArray(args)
	local box = p.renderBox(tStyles)
	local trackingCategories = p.renderTrackingCategories(args, tStyles)
	return box .. trackingCategories
end

function p.renderBox(tStyles)
	local boxArgs = {}
	if #tStyles < 1 then
		boxArgs.text = string.format('<strong class="error">%s</strong>',
	else
		local cfg = getConfig()
		local tStylesLinks = {}
		for i, ts in ipairs(tStyles) do
			local link = string.format('[[:%s]]', ts)
			local sandboxLink = nil
			local tsTitle = mw.title.new(ts)
			if tsTitle and cfg['sandbox_title'] then
				local tsSandboxTitle = mw.title.new(string.format
					'%s:%s/%s/%s', tsTitle.nsText, tsTitle.ba
				if tsSandboxTitle and tsSandboxTitle.exists then
```

```
                                        sandboxLink = format('sandboxlink', link,
                                end
                        end
                        tStylesLinks[i] = sandboxLink or link
                end
                local tStylesList = mList.makeList('bulleted', tStylesLinks)
                boxArgs.text = format(
                        mw.title.getCurrentTitle():inNamespaces(828,829) and 'hea
                        '\n' .. tStylesList
        end
        boxArgs.type = 'notice'
        boxArgs.small = true
        boxArgs.image = string.format('[[File:Farm-Fresh css add.svg|32px|alt=%s]
        return mMessageBox.main('mbox', boxArgs)
end

function p.renderTrackingCategories(args, tStyles, titleObj)
        if yesno(args.nocat) then
                return ''
        end

        local cfg = getConfig()

        local cats = {}

        -- Error category
        if #tStyles < 1 and cfg['error_category'] then
                cats[#cats + 1] = cfg['error_category']
        end

        -- TemplateStyles category
        titleObj = titleObj or mw.title.getCurrentTitle()
        if (titleObj.namespace == 10 or titleObj.namespace == 828)
                and not cfg['subpage_blacklist'][titleObj.subpageText]
        then
                local category = args.category or cfg['default_category']
                if category then
                        cats[#cats + 1] = category
                end
                if not yesno(args.noprotcat) and (cfg['protection_conflict_categ
                        local currentProt = titleObj.protectionLevels["edit"] and
                        local addedLevelCat = false
                        local addedPadlockCat = false
                        for i, ts in ipairs(tStyles) do
                                local tsTitleObj = mw.title.new(ts)
                                local tsProt = tsTitleObj.protectionLevels["edit"
                                if cfg['protection_conflict_category'] and tsProt
                                        addedLevelCat = true
                                        cats[#cats + 1] = cfg['protection_conflic
                                end
                                if cfg['padlock_pattern'] and tsProt and not added
                                        local content = tsTitleObj:getContent()
                                        if not content:find(cfg['padlock_pattern'
                                                cats[#cats + 1] = cfg['missing_pa
                                                addedPadlockCat = true
                                        end
                                end
                        end
                end
        end
        for i, cat in ipairs(cats) do
```

```
                cats[i] = string.format('[[Category:%s]]', cat)
        end
        return table.concat(cats)
end

return p
```

Modul:Uses TemplateStyles/Doku

Dies ist die Dokumentationsseite für Modul:Uses TemplateStyles

Vorlage:Lua

Implements Vorlage:Tlx.

[[Category:Module documentation pages{{#translation:}}]]

Modul:Uses TemplateStyles/config

Die Dokumentation für dieses Modul kann unter Modul:Uses TemplateStyles/config/Doku erstellt werden

```
local cfg = {} -- Don’t touch this line.

-- Subpage blacklist: these subpages will not be categorized (except for the
-- error category, which is always added if there is an error).
-- For example “Template:Foo/doc” matches the `doc = true` rule, so it will have
-- no categories. “Template:Foo” and “Template:Foo/documentation” match no rules,
-- so they *will* have categories. All rules should be in the
--   ['<subpage name>'] = true,
-- format.
cfg['subpage_blacklist'] = {
        ['doc'] = true,
        ['sandbox'] = true,
        ['sandbox2'] = true,
        ['testcases'] = true,
}

-- Sandbox title: if the stylesheet’s title is <template>/<stylesheet>.css, the
-- stylesheet’s sandbox is expected to be at <template>/<sandbox_title>/<styleshe
-- Set to nil to disable sandbox links.
cfg['sandbox_title'] = 'sandbox'

-- Error category: this category is added if the module call contains errors
-- (e.g. no stylesheet listed). A category name without namespace, or nil
-- to disable categorization (not recommended).
cfg['error_category'] = 'Uses TemplateStyles templates with errors'

-- Default category: this category is added if no custom category is specified
-- in module/template call. A category name without namespace, or nil
-- to disable categorization.
cfg['default_category'] = 'Templates using TemplateStyles'

-- Protection conflict category: this category is added if the protection level
-- of any stylesheet differs from the one of the template. A category name
-- without namespace, or nil to disable categorization (not recommended).
cfg['protection_conflict_category'] = 'Templates using TemplateStyles with a diff

-- Padlock pattern: Lua pattern to search on protected stylesheets for, or nil
-- to disable padlock check.
cfg['padlock_pattern'] = '{{pp-'

-- Missing padlock category: this category is added if a protected stylesheet
-- doesn’t contain any padlock template (specified by the above Lua pattern).
-- A category name without namespace (no nil allowed) if the pattern is not nil,
-- unused (and thus may be nil) otherwise.
cfg['missing_padlock_category'] = 'Templates using TemplateStyles without padlock

return cfg -- Don’t touch this line.
```