پودمان:Template link general

.□□□□ □□□□ [□□□□□□□□/Template link general:□□□□□□](#) □□ □□□□□□□□ □□□□□□□ □□□□ □□□□□□□□

```
-- This implements Template:Template link general and various other templates
in its family
local getArgs = require('Module:Arguments').getArgs
local p = {}

-- Is a string non-empty?
local function _ne(s)
	return s ~= nil and s ~= ""
end

local nw = mw.text.nowiki

local function addTemplate(s)
	local i, _ = s:find(':', 1, true)
	if i == nil then
		return 'Template:' .. s
	end
	local ns = s:sub(1, i - 1)
	if ns == '' or mw.site.namespaces[ns] then
		return s
	else
		return 'Template:' .. s
	end
end

local function trimTemplate(s)
	local needle = 'template:'
	if s:sub(1, needle:len()):lower() == needle then
		return s:sub(needle:len() + 1)
	else
		return s
	end
end

local function linkTitle(args)
	if _ne(args.nolink) then
		if _ne(args.subst) then
			return 'subst:' .. args['1']
		else
			return args['1']
		end
	end
```

```
	local titleObj
	local titlePart = '[['
	if args['1'] then
		-- This handles :Page and other NS
		titleObj = mw.title.new(args['1'], 'Template')
	else
		titleObj = mw.title.getCurrentTitle()
	end
	titlePart = titlePart .. (titleObj ~= nil and titleObj.fullText or
				addTemplate(args['1']))
	local textPart = args.alttext
	if not _ne(textPart) then
		if titleObj ~= nil then
			textPart = titleObj:inNamespace("Template") and
args['1'] or titleObj.fullText
		else
			-- redlink
			textPart = args['1']
		end
	end
	if _ne(args.subst) then
		-- HACK: the ns thing above is probably broken
		textPart = 'subst:' .. textPart
	end
	if _ne(args.brace) then
		textPart = nw('{{') .. textPart .. nw('}}')
	elseif _ne(args.braceinside) then
		textPart = nw('{') .. textPart .. nw('}')
	end
	titlePart = titlePart .. '|' .. textPart .. ']]'
	if _ne(args.braceinside) then
		titlePart = nw('{') .. titlePart .. nw('}')
	end
	return titlePart
end

function p.main(frame)
	local args = getArgs(frame, {
		trim = true,
		removeBlanks = false
	})
	return p._main(args)
end

function p._main(args)
	local bold = _ne(args.bold) or _ne(args.boldlink) or
_ne(args.boldname)
	local italic = _ne(args.italic) or _ne(args.italics)
	local dontBrace = _ne(args.brace) or _ne(args.braceinside)
	local code = _ne(args.code) or _ne(args.tt)
	local show_result = _ne(args._show_result)
```

```
        local expand = _ne(args._expand)
        -- Build the link part
        local titlePart = linkTitle(args)
        if bold then titlePart = "'''" .. titlePart .. "'''" end
        if _ne(args.nowrapname) then titlePart = '<span class="nowrap">' ..
titlePart .. '</span>' end
        -- Build the arguments
        local textPart = ""
        local textPartBuffer = "&#124;"
        local codeArguments = {}
        local codeArgumentsString = ""
        local i = 2
        local j = 1
        while args[i] do
                local val = args[i]
                if val ~= "" then
                        if _ne(args.nowiki) then
                                -- Unstrip nowiki tags first because calling
nw on something that already contains nowiki tags will
                                -- mangle the nowiki strip marker and result
in literal UNIQ...QINU showing up
                                val = nw(mw.text.unstripNoWiki(val))
                        end
                        local k, v = string.match(val, "(.*)=(.*)")
                        if not k then
                                codeArguments[j] = val
                                j = j + 1
                        else
                                codeArguments[k] = v
                        end
                        codeArgumentsString = codeArgumentsString ..
textPartBuffer .. val
                        if italic then
                                val = '<span style="font-style:italic;">' ..
val .. '</span>'
                        end
                        textPart = textPart .. textPartBuffer .. val
                end
                i = i + 1
        end

        -- final wrap
        local ret = titlePart .. textPart
        if not dontBrace then ret = nw('{{') .. ret .. nw('}}') end
        if _ne(args.a) then ret = nw('*') .. ' ' .. ret end
        if _ne(args.kbd) then ret = '<kbd>' .. ret .. '</kbd>' end
        if code then
                ret = '<code>' .. ret .. '</code>'
        elseif _ne(args.plaincode) then
                ret = '<code
style="border:none;background:transparent;color:inherit">' .. ret ..
```

```
'</code>'
        end
        if _ne(args.nowrap) then ret = '<span class="nowrap">' .. ret ..
'</span>' end
        --[[ Wrap as html??
        local span = mw.html.create('span')
        span:wikitext(ret)
        --]]
        if _ne(args.debug) then ret = ret .. '\n<pre>' ..
mw.text.encode(mw.dumpObject(args)) .. '</pre>' end

        if show_result then
                local result = mw.getCurrentFrame():expandTemplate{title =
addTemplate(args[1]), args = codeArguments}
                ret = ret .. " → " .. result
        end

        if expand then
                local query = mw.text.encode('{{' .. addTemplate(args[1]) ..
string.gsub(codeArgumentsString, textPartBuffer, "|") .. '}}')
                local url = mw.uri.fullUrl('special:ExpandTemplates',
'wpInput=' .. query)
                mw.log()
                ret = ret .. " [" .. tostring(url) .. "]"
        end

        return ret
end

return p
```