Inhaltsverzeichnis

1. Modul:Error 2
2. Modul:If preview 3
3. Modul:Warning 5

Modul:Error

This module creates an html message with class "error". It is a replacement for Vorlage:Tl - please see the documentation page there for usage instructions.

See also

- Module:Warning
- Module:If preview

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

local p = {}

local function _error(args)
    local tag = mw.ustring.lower(tostring(args.tag))

    -- Work out what html tag we should use.
    if not (tag == 'p' or tag == 'span' or tag == 'div') then
        tag = 'strong'
    end

    -- Generate the html.
    return tostring(mw.html.create(tag)
        :addClass('error')
        :wikitext(tostring(args.message or args[1] or error('no message specifie
    )
end

function p.error(frame)
    local args
    if type(frame.args) == 'table' then
        -- We're being called via #invoke. The args are passed through to the mo
        -- from the template page, so use the args that were passed into the temp
        args = frame.args
    else
        -- We're being called from another module or from the debug console, so a
        -- the args are passed in directly.
        args = frame
    end
    -- if the message parameter is present but blank, change it to nil so that Lu
    -- consider it false.
    if args.message == "" then
        args.message = nil
    end
    return _error(args)
end

return p
```

Modul:If preview

Vorlage:Lua Vorlage:Uses TemplateStyles

This module implements Vorlage:Tl and Vorlage:Tl. It helps templates/modules determine if they are being previewed.

Prefer implementing the template versions in other templates.

In a module to use the `main()`, you need to pass a frame table with an args table.

For the preview warning, use `_warning()`.

```
local p = {}

local cfg = mw.loadData('Module:If preview/configuration')

--[[
main

This function returns either the first argument or second argument passed to
this module, depending on whether the page is being previewed.

]]
function p.main(frame)
	if cfg.preview then
		return frame.args[1] or ''
	else
		return frame.args[2] or ''
	end
end

--[[
pmain

This function returns either the first argument or second argument passed to
this module's parent (i.e. template using this module), depending on whether it
is being previewed.

]]
function p.pmain(frame)
	return p.main(frame:getParent())
end


local function warning_text(warning)
	return mw.ustring.format(
		cfg.warning_infrastructure,
		cfg.templatestyles,
		warning
	)
end

function p._warning(args)

	local warning = args[1] and args[1]:match('^%s*(.-)%s*$') or ''
```

```
        if warning == '' then
                return warning_text(cfg.missing_warning)
        end

        if not cfg.preview then return '' end

        return warning_text(warning)
end

--[[
warning

This function returns a "preview warning", which is the first argument marked
up with HTML and some supporting text, depending on whether the page is being pre

disabled since we'll implement the template version in general

]]
--function p.warning(frame)
--        return p._warning(frame.args)
--end

--[[
warning, but for pass-through templates like {{preview warning}}
]]
function p.pwarning(frame)
        return p._warning(frame:getParent().args)
end

return p
```

Modul:Warning

This module simply unifies the formatting of all warning messages similar to Module:Error. Currently, it is plain text, but custom formatting may be applied after discussion in the talk page. Warnings are displayed above the preview when previewing an edit.

Usage

```
local warn = require('Module:Warning')
warn("Message")
warn(("TypeWarning: %s"):format(warning), level)
```

See also

- Module:Error
- Module:If preview

```
local libraryUtil = require('libraryUtil')

local wrapper = "%s" -- wikitext formatting
local msg_loc = "Lua warning in %s at line %d: %s."
local msg = "Lua warning: %s."

return function (message, level)
	libraryUtil.checkType('warn', 2, level, 'number', true)
	level = level or 1
	if level > 0 then
		local _, location = pcall(error, '', level+2)
		if location ~= '' then
			location = mw.text.split(location:sub(1,-3), ':%f[%d]')
			message = msg_loc:format(location[1], location[2], messag
		else
			message = msg:format(message)
		end
	else
		message = msg:format(message)
	end
	mw.addWarning(wrapper:format(message))
end
```