Module:Pagetype

From Blue Gold Program Wiki
The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.

Documentation for this module may be created at Module:Pagetype/doc

```
--------------------------------------------------------------------------------
--
--                                  PAGETYPE
--
--
--
--      This is a meta-module intended to replace {{pagetype}} and similar
--
--      templates. It automatically detects namespaces, and allows for a
--
--      great deal of customisation. It can easily be ported to other
--
--      wikis by changing the values in the [[Module:Pagetype/config]].
--
--
--
--------------------------------------------------------------------------------

-- Load config.
local cfg = mw.loadData('Module:Pagetype/config')

-- Load required modules.
local getArgs = require('Module:Arguments').getArgs
local yesno = require('Module:Yesno')
local nsDetectModule = require('Module:Namespace detect')
local nsDetect = nsDetectModule._main
local getParamMappings = nsDetectModule.getParamMappings
local getPageObject = nsDetectModule.getPageObject

local p = {}

local function shallowCopy(t)
        -- Makes a shallow copy of a table.
        local ret = {}
        for k, v in pairs(t) do
                ret[k] = v
        end
        return ret
end

local function checkPagetypeInput(namespace, val)
        -- Checks to see whether we need the default value for the given
namespace,
        -- and if so gets it from the pagetypes table.
        -- The yesno function returns true/false for "yes", "no", etc., and
returns
        -- val for other input.
        local ret = yesno(val, val)
```

```
        if ret and type(ret) ~= 'string' then
                ret = cfg.pagetypes[namespace]
        end
        return ret
end

local function getPagetypeFromClass(class, param, aliasTable, default)
        -- Gets the pagetype from a class specified from the first positional
        -- parameter.
        param = yesno(param, param)
        if param ~= false then -- No check if specifically disallowed.
                for _, alias in ipairs(aliasTable) do
                        if class == alias then
                                if type(param) == 'string' then
                                        return param
                                else
                                        return default
                                end
                        end
                end
        end
end

local function getNsDetectValue(args)
        -- Builds the arguments to pass to [[Module:Namespace detect]] and
returns
        -- the result.

        -- Get the default values.
        local ndArgs = {}
        local defaultns = args[cfg.defaultns]
        if defaultns == cfg.defaultnsAll then
                ndArgs = shallowCopy(cfg.pagetypes)
        else
                local defaultnsArray
                if defaultns == cfg.defaultnsExtended then
                        defaultnsArray = cfg.extendedNamespaces
                elseif defaultns == cfg.defaultnsNone then
                        defaultnsArray = {}
                else
                        defaultnsArray = cfg.defaultNamespaces
                end
                for _, namespace in ipairs(defaultnsArray) do
                        ndArgs[namespace] = cfg.pagetypes[namespace]
                end
        end

        --[[
        -- Add custom values passed in from the arguments. These overwrite
the
        -- defaults. The possible argument names are fetched from
```

```
	-- Module:Namespace detect automatically in case new namespaces are
	-- added. Although we accept namespace aliases as parameters, we only
pass
	-- the local namespace name as a parameter to Module:Namespace
detect.
	-- This means that the "image" parameter can overwrite defaults for
the
	-- File: namespace, which wouldn't work if we passed the parameters
through
	-- separately.
	--]]
	local mappings = getParamMappings()
	for ns, paramAliases in pairs(mappings) do
		-- Copy the aliases table, as # doesn't work with tables
returned from
		-- mw.loadData.
		paramAliases = shallowCopy(paramAliases)
		local paramName = paramAliases[1]
		-- Iterate backwards along the array so that any values for
the local
		-- namespace names overwrite those for namespace aliases.
		for i = #paramAliases, 1, -1 do
			local paramAlias = paramAliases[i]
			local ndArg = checkPagetypeInput(paramAlias,
args[paramAlias])
			if ndArg == false then
				-- If any arguments are false, convert them
to nil to protect
				-- against breakage by future changes to
				-- [[Module:Namespace detect]].
				ndArgs[paramName] = nil
			elseif ndArg then
				ndArgs[paramName] = ndArg
			end
		end
	end
	-- Check for disambiguation-class and N/A-class pages in mainspace.
	if ndArgs.main then
		local class = args[1]
		if type(class) == 'string' then
			-- Put in lower case so e.g. "Dab" and "dab" will
both match.
			class = mw.ustring.lower(class)
		end
		local dab = getPagetypeFromClass(
			class,
			args[cfg.dab],
			cfg.dabAliases,
			cfg.dabDefault
		)
		if dab then
```

```
                        ndArgs.main = dab
                else
                        local na = getPagetypeFromClass(
                                class,
                                args[cfg.na],
                                cfg.naAliases,
                                cfg.naDefault
                        )
                        if na then
                                ndArgs.main = na
                        end
                end
        end
        -- If there is no talk value specified, use the corresponding subject
        -- namespace for talk pages.
        if not ndArgs.talk then
                ndArgs.subjectns = true
        end
        -- Add the fallback value. This can also be customised, but it cannot
be
        -- disabled.
        local other = args[cfg.other]
        -- We will ignore true/false/nil results from yesno here, but using
it
        -- anyway for consistency.
        other = yesno(other, other)
        if type(other) == 'string' then
                ndArgs.other = other
        else
                ndArgs.other = cfg.otherDefault
        end
        -- Allow custom page values.
        ndArgs.page = args.page
        return nsDetect(ndArgs)
end

local function detectRedirects(args)
        local redirect = args[cfg.redirect]
        -- The yesno function returns true/false for "yes", "no", etc., and
returns
        -- redirect for other input.
        redirect = yesno(redirect, redirect)
        if redirect == false then
                -- Detect redirects unless they have been explicitly
disallowed with
                -- "redirect=no" or similar.
                return
        end
        local pageObject = getPageObject(args.page)
        -- If we are using subject namespaces elsewhere, do so here as well.
        if pageObject
```

```
                and not yesno(args.talk, true)
                and args[cfg.defaultns] ~= cfg.defaultnsAll
        then
                pageObject = getPageObject(
                        pageObject.subjectNsText .. ':' .. pageObject.text
                )
        end
        -- Allow custom values for redirects.
        if pageObject and pageObject.isRedirect then
                if type(redirect) == 'string' then
                        return redirect
                else
                        return cfg.redirectDefault
                end
        end
end

function p._main(args)
        local redirect = detectRedirects(args)
        if redirect then
                return redirect
        else
                return getNsDetectValue(args)
        end
end

function p.main(frame)
        local args = getArgs(frame)
        return p._main(args)
end

return p
```


Namespaces

- Module
- Discussion

Variants

This page was last edited on 19 February 2020, at 09:51.

Blue Gold Program Wiki

The wiki version of the Lessons Learnt Report of the Blue Gold program, documents the experiences of a technical assistance (TA) team working in a development project implemented by the Bangladesh Water Development Board (BWDB) and the Department of Agricultural Extension (DAE) over an eight+ year period from March 2013 to December 2021. The wiki lessons learnt report (LLR) is intended to complement the BWDB and DAE project completion reports (PCRs), with the

aim of recording lessons learnt for use in the design and implementation of future interventions in the coastal zone.