Inhaltsverzeichnis

1. Modul:Namespace detect/data/Doku 2
2. Modul:Category handler/shared 3
3. Modul:Namespace detect 4
4. Modul:Namespace detect/data 10

Modul:Namespace detect/data/Doku

Dies ist die Dokumentationsseite für Modul:Namespace detect/data

This is a data page for Module:Namespace detect and Module:Category handler/shared. It is loaded by the main module using mw.loadData, which means it is only processed once per page rather than once per #invoke.

Modul:Category handler/shared

Usage

{{#invoke:Category handler|*function_name*}}

```
-- This module contains shared functions used by [[Module:Category handler]]
-- and its submodules.

local p = {}

function p.matchesBlacklist(page, blacklist)
	for i, pattern in ipairs(blacklist) do
		local match = mw.ustring.match(page, pattern)
		if match then
			return true
		end
	end
	return false
end

function p.getParamMappings(useLoadData)
	local dataPage = 'Module:Namespace detect/data'
	if useLoadData then
		return mw.loadData(dataPage).mappings
	else
		return require(dataPage).mappings
	end
end

function p.getNamespaceParameters(titleObj, mappings)
	-- We don't use title.nsText for the namespace name because it adds
	-- underscores.
	local mappingsKey
	if titleObj.isTalkPage then
		mappingsKey = 'talk'
	else
		mappingsKey = mw.site.namespaces[titleObj.namespace].name
	end
	mappingsKey = mw.ustring.lower(mappingsKey)
	return mappings[mappingsKey] or {}
end

return p
```

Modul:Namespace detect

This module allows you to output different text depending on the namespace that a given page is in. It is a Lua implementation of the Vorlage:Tl template, with a few improvements: all namespaces and all namespace aliases are supported, and namespace names are detected automatically for the local wiki.

Inhaltsverzeichnis

1 Usage 4
2 Parameters 4
2.1 Namespace parameters 5
3 Table function 6
4 Porting to different wikis 6
5 Technical details 6

Usage

```
{{#invoke: Namespace detect | main
| page              = <!-- page to detect namespace for, if not the current page
| main              = <!-- text to return for the main namespace -->
| talk              = <!-- text to return for talk namespaces -->

<!-- text to return for specific subject namespaces -->
| portal            =
| category          =
| user                    =
| wikipedia         =
| mediawiki         =
| book              =
| timedtext         =
| template          =
| special           =
| media             =
| file              =
| image             =
| help                    =
| module            =

| other             = <!-- text to return for unspecified namespaces -->
| demospace         = <!-- namespace to display text for -->

| subjectns         = <!-- set to "yes" to treat talk pages as the corresponding
}}
```

Parameters

- **main** - text to return if the page is in the main namespace.
- **talk** - text to return if the page is in a talk namespace. This can be any talk namespace - it will match any of "Talk:", "Wikipedia talk:", "User talk:", etc.

- Subject namespace parameters, e.g. **wikipedia**, **user**, **file**... - the text to return if the page is in the corresponding namespace. This module accepts all subject namespaces as parameters, including namespace aliases and virtual namespaces. See below for a list of supported values.
- **other** - text to return if no parameters for the page's namespace were specified. This text is also returned if Vorlage:Para is set to an invalid namespace value.
- **subjectns** - if on a talk page, use the corresponding subject page. Can be set with values of "yes", "y", "true" or "1".
- **demopage** - specifies a page to detect the namespace of. If not specified, and if the Vorlage:Para parameter is not set, then the module uses the current page.
- **demospace** - force the module to behave as if the page was in the specified namespace. Often used for demonstrations.

Namespace parameters

Possible values for subject namespace parameters are as follows:

Namespace	Aliases
main	
benutzer	user, benutzerin
projekt	project
datei	file, bild, image
mediawiki	
vorlage	template
hilfe	help
kategorie	category
attribut	property
formular	form
konzept	concept
smw/schema	smw/schema
rule	
widget	
campaign	
timedtext	
modul	module
blog	
socialentity	
gadget	
gadget-definition	gadget definition
copikiblog	

Namespace	Aliases
deutsch	
english	
français	
italiano	
español	
nederlands	
slovenská	
dansk	
polska	
suomi	

Table function

Use the following to display a table with the different possible namespace parameters:

```
{{#invoke:Namespace detect|table|talk=yes}}
```

To include the parameter for talk namespaces, use Vorlage:Para.

Porting to different wikis

This module is designed to be portable. To use it on a different wiki, all you need to do is to change the values in Module:Namespace detect/config. Instructions are available on that page.

Technical details

The module uses a data page at Module:Namespace detect/data. This page is loaded with mw.loadData, which means it is processed once per page rather than once per #invoke. This was done for performance reasons.

```
--[[
--------------------------------------------------------------------------------
--                                                                            --
--                            NAMESPACE DETECT                                --
--                                                                            --
-- This module implements the {{namespace detect}} template in Lua, with a    --
-- few improvements: all namespaces and all namespace aliases are supported,  --
-- and namespace names are detected automatically for the local wiki. The     --
-- module can also use the corresponding subject namespace value if it is     --
-- used on a talk page. Parameter names can be configured for different wikis --
-- by altering the values in the "cfg" table in                               --
-- Module:Namespace detect/config.                                            --
--                                                                            --
--------------------------------------------------------------------------------
--]]
```

```
local data = mw.loadData('Module:Namespace detect/data')
local argKeys = data.argKeys
local cfg = data.cfg
local mappings = data.mappings

local yesno = require('Module:Yesno')
local mArguments -- Lazily initialise Module:Arguments
local mTableTools -- Lazily initilalise Module:TableTools
local ustringLower = mw.ustring.lower

local p = {}

local function fetchValue(t1, t2)
	-- Fetches a value from the table t1 for the first key in array t2 where
	-- a non-nil value of t1 exists.
	for i, key in ipairs(t2) do
		local value = t1[key]
		if value ~= nil then
			return value
		end
	end
	return nil
end

local function equalsArrayValue(t, value)
	-- Returns true if value equals a value in the array t. Otherwise
	-- returns false.
	for i, arrayValue in ipairs(t) do
		if value == arrayValue then
			return true
		end
	end
	return false
end

function p.getPageObject(page)
	-- Get the page object, passing the function through pcall in case of
	-- errors, e.g. being over the expensive function count limit.
	if page then
		local success, pageObject = pcall(mw.title.new, page)
		if success then
			return pageObject
		else
			return nil
		end
	else
		return mw.title.getCurrentTitle()
	end
end

-- Provided for backward compatibility with other modules
function p.getParamMappings()
	return mappings
end

local function getNamespace(args)
	-- This function gets the namespace name from the page object.
	local page = fetchValue(args, argKeys.demopage)
	if page == '' then
		page = nil
	end
	local demospace = fetchValue(args, argKeys.demospace)
	if demospace == '' then
```

```
                demospace = nil
        end
        local subjectns = fetchValue(args, argKeys.subjectns)
        local ret
        if demospace then
                -- Handle "demospace = main" properly.
                if equalsArrayValue(argKeys.main, ustringLower(demospace)) then
                        ret = mw.site.namespaces[0].name
                else
                        ret = demospace
                end
        else
                local pageObject = p.getPageObject(page)
                if pageObject then
                        if pageObject.isTalkPage then
                                -- Get the subject namespace if the option is set
                                -- otherwise use "talk".
                                if yesno(subjectns) then
                                        ret = mw.site.namespaces[pageObject.names
                                else
                                        ret = 'talk'
                                end
                        else
                                ret = pageObject.nsText
                        end
                else
                        return nil -- return nil if the page object doesn't exist
                end
        end
        ret = ret:gsub('_', ' ')
        return ustringLower(ret)
end

function p._main(args)
        -- Check the parameters stored in the mappings table for any matches.
        local namespace = getNamespace(args) or 'other' -- "other" avoids nil tab
        local params = mappings[namespace] or {}
        local ret = fetchValue(args, params)
        --[[
        -- If there were no matches, return parameters for other namespaces.
        -- This happens if there was no text specified for the namespace that
        -- was detected or if the demospace parameter is not a valid
        -- namespace. Note that the parameter for the detected namespace must be
        -- completely absent for this to happen, not merely blank.
        --]]
        if ret == nil then
                ret = fetchValue(args, argKeys.other)
        end
        return ret
end

function p.main(frame)
        mArguments = require('Module:Arguments')
        local args = mArguments.getArgs(frame, {removeBlanks = false})
        local ret = p._main(args)
        return ret or ''
end

function p.table(frame)
        --[[
        -- Create a wikitable of all subject namespace parameters, for
        -- documentation purposes. The talk parameter is optional, in case it
        -- needs to be excluded in the documentation.
        --]]
```

```
        -- Load modules and initialise variables.
        mTableTools = require('Module:TableTools')
        local namespaces = mw.site.namespaces
        local cfg = data.cfg
        local useTalk = type(frame) == 'table'
                and type(frame.args) == 'table'
                and yesno(frame.args.talk) -- Whether to use the talk parameter.

        -- Get the header names.
        local function checkValue(value, default)
                if type(value) == 'string' then
                        return value
                else
                        return default
                end
        end
        local nsHeader = checkValue(cfg.wikitableNamespaceHeader, 'Namespace')
        local aliasesHeader = checkValue(cfg.wikitableAliasesHeader, 'Aliases')

        -- Put the namespaces in order.
        local mappingsOrdered = {}
        for nsname, params in pairs(mappings) do
                if useTalk or nsname ~= 'talk' then
                        local nsid = namespaces[nsname].id
                        -- Add 1, as the array must start with 1; nsid 0 would be
                        nsid = nsid + 1
                        mappingsOrdered[nsid] = params
                end
        end
        mappingsOrdered = mTableTools.compressSparseArray(mappingsOrdered)

        -- Build the table.
        local ret = '{| class="wikitable"'
                .. '\n|-'
                .. '\n! ' .. nsHeader
                .. '\n! ' .. aliasesHeader
        for i, params in ipairs(mappingsOrdered) do
                for j, param in ipairs(params) do
                        if j == 1 then
                                ret = ret .. '\n|-'
                                        .. '\n| <code>' .. param .. '</code>'
                                        .. '\n| '
                        elseif j == 2 then
                                ret = ret .. '<code>' .. param .. '</code>'
                        else
                                ret = ret .. ', <code>' .. param .. '</code>'
                        end
                end
        end
        ret = ret .. '\n|-'
                .. '\n|}'
        return ret
end

return p
```

Modul:Namespace detect/data

This is a data page for Module:Namespace detect and Module:Category handler/shared. It is loaded by the main module using mw.loadData, which means it is only processed once per page rather than once per #invoke.

```
--------------------------------------------------------------------------------
--                            Namespace detect data                           --
-- This module holds data for [[Module:Namespace detect]] to be loaded per    --
-- page, rather than per #invoke, for performance reasons.                    --
--------------------------------------------------------------------------------

local cfg = require('Module:Namespace detect/config')

local function addKey(t, key, defaultKey)
	if key ~= defaultKey then
		t[#t + 1] = key
	end
end

-- Get a table of parameters to query for each default parameter name.
-- This allows wikis to customise parameter names in the cfg table while
-- ensuring that default parameter names will always work. The cfg table
-- values can be added as a string, or as an array of strings.

local defaultKeys = {
	'main',
	'talk',
	'other',
	'subjectns',
	'demospace',
	'demopage'
}

local argKeys = {}
for i, defaultKey in ipairs(defaultKeys) do
	argKeys[defaultKey] = {defaultKey}
end

for defaultKey, t in pairs(argKeys) do
	local cfgValue = cfg[defaultKey]
	local cfgValueType = type(cfgValue)
	if cfgValueType == 'string' then
		addKey(t, cfgValue, defaultKey)
	elseif cfgValueType == 'table' then
		for i, key in ipairs(cfgValue) do
			addKey(t, key, defaultKey)
		end
	end
	cfg[defaultKey] = nil -- Free the cfg value as we don't need it any more.
end

local function getParamMappings()
	--[[
	-- Returns a table of how parameter names map to namespace names. The key
	-- are the actual namespace names, in lower case, and the values are the
	-- possible parameter names for that namespace, also in lower case. The
	-- table entries are structured like this:
	-- {
```

```
        --   [''] = {'main'},
        --   ['wikipedia'] = {'wikipedia', 'project', 'wp'},
        --   ...
        -- }
        --]]
        local mappings = {}
        local mainNsName = mw.site.subjectNamespaces[0].name
        mainNsName = mw.ustring.lower(mainNsName)
        mappings[mainNsName] = mw.clone(argKeys.main)
        mappings['talk'] = mw.clone(argKeys.talk)
        for nsid, ns in pairs(mw.site.subjectNamespaces) do
                if nsid ~= 0 then -- Exclude main namespace.
                        local nsname = mw.ustring.lower(ns.name)
                        local canonicalName = mw.ustring.lower(ns.canonicalName)
                        mappings[nsname] = {nsname}
                        if canonicalName ~= nsname then
                                table.insert(mappings[nsname], canonicalName)
                        end
                        for _, alias in ipairs(ns.aliases) do
                                table.insert(mappings[nsname], mw.ustring.lower(a
                        end
                end
        end
        return mappings
end

return {
        argKeys = argKeys,
        cfg = cfg,
        mappings = getParamMappings()
}
```