https://fa.wikimarefat.ir/view/%D9%BE%D9%88%D8%AF%D9%85%D8%A7%D9%86:Labelled_list_hatnote

پودمان:Labelled list hatnote



این پودمان چندین تابع کاربردی را فراهم می‌کند که کار پیاده‌سازی سرنویس‌هایی که از قالب یک برچسب پیش از فهرستی از صفحه‌ها برخوردار هستند را آسان می‌کنند. نمونه‌ای از سرنویس‌های مذکور:

□□□□□: □□□□ □□ □ □

استفاده

labelledList

برای پیاده‌سازی بیشتر الگوهایی از این دست، درخواست برای اجرای تابع labelledList() کفایت می‌کند:

```
{{#درخواست:labelledList|Labelled list hatnote|برچسب سراسری}}
```

یا

```
{{#درخواست:labelledList|Labelled list hatnote|برچسب مفرد|برچسب جمع}}
```

برای مثال، وارد کردن «همچنین ببینید» به‌جای «برچسب سراسری» در نمونهٔ نخست، عملکردی یکسان با {{همچنین ببینید}} را ارائه می‌دهد، و وارد کردن «مقالهٔ اصلی» و «مقاله‌های اصلی» به‌جای «برچسب مفرد» و «برچسب جمع» منجر به عملکردی یکسان با الگوی {{اصلی}} (در فضای نام مقاله) می‌شود.

اگر برچسب‌های سوم و چهارم نیز وارد شده باشند، در مواردی به‌کار خواهند رفت که هریک از صفحه‌های هدف در خارج از فضای نام مقاله باشند؛ بنابراین، برای مثال {{اصلی}} می‌تواند به شکل زیر پیاده‌سازی شود:

```
{{#درخواست:labelledList|Labelled list hatnote|مقالهٔ اصلی|مقاله‌های اصلی|صفحهٔ اصلی|صفحه‌های اصلی}}
```

preprocessDisplays

تابع preprocessDisplays() فهرستی خام از آرگومان‌ها را می‌پذیرد و هرگونه آرگومان نمایشی را با آن ترکیب می‌کند. برای مثال، {{همچنین ببینید|۱|ب۱=یک}} در ابتدا فهرست آرگومان‌های {'۱', ['ب۱'] = 'یک'} را دارد؛ این فهرست، در حالت ترکیب شده به‌شکل {'1|یک'} در فهرست جدید قرار می‌گیرد. این تابع از نام‌پوشی دستی (یعنی {{همچنین ببینید|۱{{!}}۲{{!}}ب۱=یک}} ← {'۱|یک'}) چشم‌پوشی می‌کند و در صورت صرف نظر از یک پارامتر یا خالی گذاشتن آن، آرایه‌های پراکنده را فشرده‌سازی می‌کند.

نمونه:

```
local mLabelledList = require('Module:Labelled list hatnote')
local pages = mLabelledList.preprocessDisplays(args)
```

labelledList_

برای پودمان‌هایی که همزمان با استفاده از این پودمان، نیازمند اعمال تغییرات جزئی در عملکرد آن هستند، تابع

_labelledList() تا حدودی انعطاف‌پذیرتر است. این تابع سه پارامتر را می‌پذیرد:

1. فهرستی از صفحه‌ها، ترجیحاً پیش-پردازش‌شده و فشرده‌سازی‌شده توسط preprocessDisplays
2. جدولی از برچسب‌ها، که آیتم اول آن حاوی برچسب مفرد یا برچسب سراسری باشد، و پارامتر دوم آن نیز جاوی برچسب جمع یا رونوشتی از پارامتر اول باشد.
3. جدولی از گزینه‌ها، ترجیحاً حاوی موارد زیر:
   - یک رشتهٔ template حاوی عنوان کامل الگو. به‌طور پیش‌فرض بر روی عنوان این پودمان تنظیم شده‌است.
   - یک رشتهٔ category (یا nil) به‌شکلی توسط makeWikitextError در پودمان:Hatnote پذیرفته می‌شود، برای غیرفعال‌سازی اختیاری رده‌های خطا
   - یک رشتهٔ selfref (یا nil) به‌شکلی که توسط _hatnote برای فعال‌سازی گزینهٔ خودارجاع پذیرفته می‌شود

نمونه:

```
local mLabelledList = require('Module:Labelled list hatnote')
return mLabelledList._labelledList(pages, labels, options)
```

خطاها

در صورتی که نام هیچ صفحه‌های در قالب پارامترهای الگو وارد نشده باشد، این پودمان باعث می‌شود تا الگوهای مبتنی بر آن یک پیام خطا تولید کنند. در حالت عادی، این پیام باید به بخش «خطاها» در صفحهٔ مستندات الگوها هدایت شود. با این حال، اگر الگوهای مزبور از این پودمان با تابع _labelledList() استفاده کنند و در جدول گزینه‌های خود آیتم template را ارائه ندهند، ان پیام خطا به‌طور پیش‌فرض به همین بخش که در حال مطالعه‌اش هستید، هدایت می‌شود. آن خطا را می‌توان با وارد کردن نام دست کم یک صفحهٔ معتبر به‌عنوان پارامتر الگوی مورد بحث، برطرف کرد؛ مشکل موجود در الگو را نیز می‌توان با وارد کردن مقدار به آیتم template در جدول options تابع _labelledList() حل کرد.

```
------------------------------------------------------------------------
---
--                            Labelled list
--
--
--
-- This module does the core work of creating a hatnote composed of a list
--
-- prefixed by a colon-terminated label, i.e. "LABEL: [andList of pages]",
--
-- for {{see also}} and similar templates.
--
------------------------------------------------------------------------
---

local mHatnote = require('Module:Hatnote')
local mHatlist = require('Module:Hatnote list')
local numConv --initialize lazily
local mArguments --initialize lazily
local yesno --initialize lazily
```

```
local p = {}

-- Defaults global to this module
local defaults = {
	label = 'همچنین نگاه کنید به', --Final fallback for label argument
	labelForm = '%s: %s',
	prefixes = {'label', 'label ', 'l'},
	faPrefixes = {'ب' ,' برچسب','برچسب' ,'ع' ,' عنوان' ,'عنوان'},
	template = 'Module:Labelled list hatnote'
}

-- Localizable message strings
local msg = {
	errorSuffix = 'خطا#',
	noInputWarning = 'نام صفحه مشخص نشده‌است',
	noOutputWarning =
		"'''[[%s]] – بدون خروجی: هیچ‌یک از صفحه‌های هدف موجود
نیستند.'''"
}

-- Helper function that pre-combines display parameters into page arguments.
-- Also compresses sparse arrays, as a desirable side-effect.
function p.preprocessDisplays (args, prefixes, faPrefixes)
	-- Prefixes specify which parameters, in order, to check for display
options
	-- They each have numbers auto-appended, e.g. 'label1', 'label 1', &
'l1'
	numConv = require('Module:Numeral converter').convert
	prefixes = prefixes or defaults.prefixes
	faPrefixes = faPrefixes or defaults.faPrefixes
	local indices = {}
	local sparsePages = {}
	for k, v in pairs(args) do
		if type(k) == 'number' then
			indices[#indices + 1] = k
			local display
			for i = 1, #prefixes do
				display = args[prefixes[i] .. k]
				if not display then
					for j = 1, #faPrefixes do
						display = args[faPrefixes[j]
.. numConv('fa', k)]
						if display then
							break
						end
					end
				else
					break
				end
			end
			sparsePages[k] = display and
```

```
                                mw.ustring.format('%s|%s', mw.ustring.gsub(v,
'|.*$', ''), display) or v
                end
        end
        table.sort(indices)
        local pages = {}
        for k, v in ipairs(indices) do pages[#pages + 1] = sparsePages[v] end
        return pages
end

--Helper function to get a page target from a processed page string
--e.g. "Page|Label" → "Page" or "Target" → "Target"
local function getTarget(pagename)
        local pipe = mw.ustring.find(pagename, '|')
        return mw.ustring.sub(pagename, 0, pipe and pipe - 1 or nil)
end

-- Produces a labelled pages-list hatnote.
-- The main frame (template definition) takes 1 or 2 arguments, for a
singular
-- and (optionally) plural label respectively:
-- * {{#invoke:Labelled list hatnote|labelledList|Singular label|Plural
label}}
-- The resulting template takes pagename & label parameters normally.
function p.labelledList (frame)
        mArguments = require('Module:Arguments')
        yesno = require('Module:Yesno')
        local labels = {frame.args[1] or defaults.label}
        labels[2] = frame.args[2] or labels[1]
        labels[3] = frame.args[3] --no defaulting
        labels[4] = frame.args[4] --no defaulting
        local template = frame:getParent():getTitle()
        local args = mArguments.getArgs(frame, {parentOnly = true})
                if args['رده'] then args.category = args['رده'] end
                if args['خودارجاع'] then args.selfref = args['خودارجاع'] end
                if args['فضای نام'] then args.namespace = args['فضای نام']
end
        local pages = p.preprocessDisplays(args)
        local options = {
                category = yesno(args.category),
                extraclasses = frame.args.extraclasses or frame.args['کلاس
اضافی'],
                ifexists = yesno(frame.args.ifexists or
frame.args['اگرموجود']),
                namespace = frame.args.namespace or frame.args['فضای نام'] or
args.namespace,
                selfref = yesno(frame.args.selfref or frame.args['خودارجاع']
or args.selfref),
                template = template
        }
        return p._labelledList(pages, labels, options)
```

```
end

function p._labelledList (pages, labels, options)
	if options.ifexists then
		for k = #pages, 1, -1 do --iterate backwards to allow smooth
removals
			local v = pages[k]
			local title = mw.title.new(getTarget(v), namespace)
			if (v == '') or title == nil or not title.exists then
				table.remove(pages, k)
			end
		end
	end
	labels = labels or {}
	label = (#pages == 1 and labels[1] or labels[2]) or defaults.label
	for k, v in pairs(pages) do
		if mHatnote.findNamespaceId(v) ~= 0 then
			label =
				(
					#pages == 1 and
					(labels[3] or labels[1] or
defaults.label) or
					(labels[4] or labels[2] or
defaults.label)
				) or defaults.label
		end
	end
	if #pages == 0 then
		if options.ifexists then
			mw.addWarning(
				mw.ustring.format(
					msg.noOutputWarning, options.template
or defaults.template
				)
			)
			return ''
		else
			return mHatnote.makeWikitextError(
				msg.noInputWarning,
				(options.template or defaults.template) ..
msg.errorSuffix,
				options.category
			)
		end
	end
	local text = mw.ustring.format(
		options.labelForm or defaults.labelForm,
		label,
		mHatlist.andList(pages, true)
	)
	local hnOptions = {
```

```
                extraclasses = options.extraclasses,
                selfref = options.selfref
        }
        return mHatnote._hatnote(text, hnOptions)
end

return p
```