پودمان:Shortcut

.□□□□ □□□□ [□□□□□□□□/Shortcut:□□□□□□□](#) □□ □□□□□□□□ □□□□□□□ □□□□ □□□□□□□□

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

-- Set constants
local CONFIG_MODULE = 'Module:Shortcut/config'

-- Load required modules
local checkType = require('libraryUtil').checkType
local yesno = require('Module:Yesno')

local p = {}

local function message(msg, ...)
	return mw.message.newRawMessage(msg, ...):plain()
end

local function makeCategoryLink(cat)
	return mw.ustring.format('[[%s:%s]]', mw.site.namespaces[14].name,
cat)
end

function p._main(shortcuts, options, frame, cfg)
	checkType('_main', 1, shortcuts, 'table')
	checkType('_main', 2, options, 'table', true)
	options = options or {}
	frame = frame or mw.getCurrentFrame()
	cfg = cfg or mw.loadData(CONFIG_MODULE)
	local templateMode = (options.template or options['الگو']) and
(yesno(options.template) or yesno(options['الگو']))
	local redirectMode = (options.redirect or options['تغییرمسیر']) and
(yesno(options.redirect) or yesno(options['تغییرمسیر']))
	local isCategorized = (not options.category) or (not options['رده'])
or (yesno(options.category) ~= false) or (yesno(options['رده']) ~= false)

	-- Validate shortcuts
	for i, shortcut in ipairs(shortcuts) do
		if type(shortcut) ~= 'string' or #shortcut < 1 then
			error(message(cfg['invalid-shortcut-error'], i), 2)
		end
	end

	-- Make the list items. These are the shortcuts plus any extra lines
such
```

```
        -- as options.msg.
        local listItems = {}
        for i, shortcut in ipairs(shortcuts) do
                local templatePath, prefix
                if templateMode then
                        -- Namespace detection
                        local titleObj = mw.title.new(shortcut, 10)
                        if titleObj.namespace == 10 then
                                templatePath = titleObj.fullText
                        else
                                templatePath = shortcut
                        end
                        prefix = options['pre' .. i] or options.pre or
options['پیشوند'] or ''
                end
                if (options.target or options['مقصد']) and
(yesno(options.target) or yesno(options['مقصد'])) then
                        listItems[i] = templateMode
                                and
mw.ustring.format("&#123;&#123;%s[[%s|%s]]&#125;&#125;", prefix,
templatePath, shortcut)
                                or mw.ustring.format("[[%s]]", shortcut)
                else
                        listItems[i] = frame:expandTemplate{
                                title = 'تغییرمسیر ممنوع',
                                args = templateMode and {templatePath,
shortcut} or {shortcut, shortcut}
                        }
                        if templateMode then
                                listItems[i] =
mw.ustring.format("&#123;&#123;%s%s&#125;&#125;", prefix, listItems[i])
                        end
                end
        end
        table.insert(listItems, (options.msg or options['پیام']))

        -- Return an error if we have nothing to display
        if #listItems < 1 then
                local msg = cfg['no-content-error']
                msg = mw.ustring.format('<strong class="error">%s</strong>',
msg)
                if isCategorized and cfg['no-content-error-category'] then
                        msg = msg .. makeCategoryLink(cfg['no-content-error-
category'])
                end
                return msg
        end

        local root = mw.html.create()
        root:wikitext(frame:extensionTag{ name = 'templatestyles', args = {
src = 'میانبر/styles.css'} })
```

```
	-- Anchors
	local anchorDiv = root
		:tag('div')
			:addClass('module-shortcutanchordiv')
	for i, shortcut in ipairs(shortcuts) do
		local anchor = mw.uri.anchorEncode(shortcut)
		anchorDiv:tag('span'):attr('id', anchor)
	end

	-- Shortcut heading
	local shortcutHeading
	do
		local nShortcuts = #shortcuts
		if nShortcuts > 0 then
			local headingMsg = options['shortcut-heading'] or
				redirectMode and cfg['redirect-heading'] or
				cfg['shortcut-heading']
			shortcutHeading = message(headingMsg, nShortcuts)
			shortcutHeading = frame:preprocess(shortcutHeading)
		end
	end

	-- Shortcut box
	local shortcutList = root
		:tag('div')
			:addClass('module-shortcutboxplain plainlist
noprint')
			:attr('role', 'note')
	if (options.float and (options.float:lower() == 'right' or
options.float == 'راست')) or (options['شناور'] and (options['شناور']:lower()
== 'right' or options['راست' == ['شناور')) then
		shortcutList:addClass('module-shortcutboxright')
	end
	if (options.clear and options.clear ~= '') or (options['پاککن'] and
options['پاککن'] ~= '') then
		shortcutList:css('clear', (options.clear or
options['پاککن']))
	end
	if shortcutHeading then
		shortcutList
			:tag('div')
				:addClass('module-shortcutlist')
				:wikitext(shortcutHeading)
	end
	local ubl = require('Module:List').unbulleted(listItems)
	shortcutList:wikitext(ubl)
	return tostring(root)
end

function p.main(frame)
	local args = require('Module:Arguments').getArgs(frame)
```

```
	-- Separate shortcuts from options
	local shortcuts, options = {}, {}
	for k, v in pairs(args) do
		if type(k) == 'number' then
			shortcuts[k] = v
		else
			options[k] = v
		end
	end

	-- Compress the shortcut array, which may contain nils.
	local function compressArray(t)
		local nums, ret = {}, {}
		for k in pairs(t) do
			nums[#nums + 1] = k
		end
		table.sort(nums)
		for i, num in ipairs(nums) do
			ret[i] = t[num]
		end
		return ret
	end
	shortcuts = compressArray(shortcuts)

	return p._main(shortcuts, options, frame)
end

return p
```