پودمان:TNT

.□□□□ □□□□ □□□□□□□□/TNT:□□□□□□ □□ □□□□□□□□ □□□□□□ □□□ □□□□□□□□

```
--
-- INTRO:   (!!! DO NOT RENAME THIS PAGE !!!)
--    This module allows any template or module to be copy/pasted between
--    wikis without any translation changes. All translation text is stored
--    in the global  Data:*.tab  pages on Commons, and used everywhere.
--
-- SEE:   https://www.mediawiki.org/wiki/Multilingual_Templates_and_Modules
--
-- ATTENTION:
--    Please do NOT rename this module - it has to be identical on all wikis.
--    This code is maintained at https://www.mediawiki.org/wiki/Module:TNT
--    Please do not modify it anywhere else, as it may get copied and
override your changes.
--    Suggestions can be made at
https://www.mediawiki.org/wiki/Module_talk:TNT
--
-- DESCRIPTION:
--    The "msg" function uses a Commons dataset to translate a message
--    with a given key (e.g. source-table), plus optional arguments
--    to the wiki markup in the current content language.
--    Use lang=xx to set language.  Example:
--
--    {{#invoke:TNT | msg
--     | I18n/Template:Graphs.tab  <!--
https://commons.wikimedia.org/wiki/Data:I18n/Template:Graphs.tab -->
--     | source-table              <!-- uses a translation message with id =
"source-table" -->
--     | param1 }}                 <!-- optional parameter -->
--
--
--    The "doc" function will generate the <templatedata> parameter
documentation for templates.
--    This way all template parameters can be stored and localized in a
single Commons dataset.
--    NOTE: "doc" assumes that all documentation is located in
Data:Templatedata/* on Commons.
--
--    {{#invoke:TNT | doc | Graph:Lines }}
--        uses
https://commons.wikimedia.org/wiki/Data:Templatedata/Graph:Lines.tab
--        if the current page is Template:Graph:Lines/doc
--
```

```
local p = {}
local i18nDataset = 'I18n/Module:TNT.tab'

-- Forward declaration of the local functions
local sanitizeDataset, loadData, link, formatMessage

function p.msg(frame)
        local dataset, id
        local params = {}
        local lang = nil
        for k, v in pairs(frame.args) do
                if k == 1 then
                        dataset = mw.text.trim(v)
                elseif k == 2 then
                        id = mw.text.trim(v)
                elseif type(k) == 'number' then
                        table.insert(params, mw.text.trim(v))
                elseif k == 'lang' and v ~= '_' then
                        lang = mw.text.trim(v)
                end
        end
        return formatMessage(dataset, id, params, lang)
end

-- Identical to p.msg() above, but used from other lua modules
-- Parameters:  name of dataset, message key, optional arguments
-- Example with 2 params:  format('I18n/Module:TNT', 'error_bad_msgkey', 'my-
key', 'my-dataset')
function p.format(dataset, key, ...)
        local checkType = require('libraryUtil').checkType
        checkType('format', 1, dataset, 'string')
        checkType('format', 2, key, 'string')
        return formatMessage(dataset, key, {...})
end


-- Identical to p.msg() above, but used from other lua modules with the
language param
-- Parameters:  language code, name of dataset, message key, optional
arguments
-- Example with 2 params:  formatInLanguage('es', I18n/Module:TNT',
'error_bad_msgkey', 'my-key', 'my-dataset')
function p.formatInLanguage(lang, dataset, key, ...)
        local checkType = require('libraryUtil').checkType
        checkType('formatInLanguage', 1, lang, 'string')
        checkType('formatInLanguage', 2, dataset, 'string')
        checkType('formatInLanguage', 3, key, 'string')
        return formatMessage(dataset, key, {...}, lang)
end

-- Obsolete function that adds a 'c:' prefix to the first param.
```

```
-- "Sandbox/Sample.tab" -> 'c:Data:Sandbox/Sample.tab'
function p.link(frame)
	return link(frame.args[1])
end

function p.doc(frame)
	local dataset = 'Templatedata/' .. sanitizeDataset(frame.args[1])
	return frame:extensionTag('templatedata', p.getTemplateData(dataset))
..
		formatMessage(i18nDataset, 'edit_doc', {link(dataset)})
end

function p.getTemplateData(dataset)
	-- TODO: add '_' parameter once lua starts reindexing properly for
"all" languages
	local data = loadData(dataset)
	local names = {}
	for _, field in pairs(data.schema.fields) do
		table.insert(names, field.name)
	end

	local params = {}
	local paramOrder = {}
	for _, row in pairs(data.data) do
		local newVal = {}
		local name = nil
		for pos, val in pairs(row) do
			local columnName = names[pos]
			if columnName == 'name' then
				name = val
			else
				newVal[columnName] = val
			end
		end
		if name then
			params[name] = newVal
			table.insert(paramOrder, name)
		end
	end

	-- Work around json encoding treating {"1":{...}} as an [{...}]
	params['zzz123']=''

	local json = mw.text.jsonEncode({
		params=params,
		paramOrder=paramOrder,
		description=data.description
	})

	json = string.gsub(json,'"zzz123":"",?', "")
```

```
        return json
end

-- Local functions

sanitizeDataset = function(dataset)
        if not dataset then
                return nil
        end
        dataset = mw.text.trim(dataset)
        if dataset == '' then
                return nil
        elseif string.sub(dataset,-4) ~= '.tab' then
                return dataset .. '.tab'
        else
                return dataset
        end
end

loadData = function(dataset, lang)
        dataset = sanitizeDataset(dataset)
        if not dataset then
                error(formatMessage(i18nDataset, 'error_no_dataset', {}))
        end

        -- Give helpful error to thirdparties who try and copy this module.
        if not mw.ext or not mw.ext.data or not mw.ext.data.get then
                error('افزونهٔ JsonConfig ناموجود است؛ ناتوان در بارگیری
https://commons.wikimedia.org/wiki/Data:' .. dataset)
        end

        local data = mw.ext.data.get(dataset, lang)

        if data == false then
                if dataset == i18nDataset then
                        -- Prevent cyclical calls
                        error('مجموعه دادهٔ ' .. i18nDataset .. ' در
ویکی‌انبار موجود نیست')
                else
                        error(formatMessage(i18nDataset, 'error_bad_dataset',
{link(dataset)}))
                end
        end
        return data
end

-- Given a dataset name, convert it to a title with the 'commons:data:'
prefix
link = function(dataset)
        return 'c:Data:' .. mw.text.trim(dataset or '')
end
```

```
formatMessage = function(dataset, key, params, lang)
	for _, row in pairs(loadData(dataset, lang).data) do
		local id, msg = unpack(row)
		if id == key then
			local result = mw.message.newRawMessage(msg,
unpack(params or {}))
			return result:plain()
		end
	end
	if dataset == i18nDataset then
		-- Prevent cyclical calls
		error('کلید پیام «' .. key .. '» نامعتبر است')
	else
		error(formatMessage(i18nDataset, 'error_bad_msgkey', {key,
link(dataset)}))
	end
end

return p
```