Inhaltsverzeichnis

Modul:Check for unknown parameters

Vorlage:Lua

This module may be appended to a template to check for uses of unknown parameters.

Inhaltsverzeichnis

Usage

Basic usage

```
{{#invoke:check for unknown parameters|check
|unknown=[[Category:Some tracking category]]
|arg1|arg2|arg3|argN}}
```

or to sort the entries in the tracking category by parameter with a preview error message

```
{{#invoke:check for unknown parameters|check
|unknown=[[Category:Some tracking category|_VALUE_]]
|preview=unknown parameter "_VALUE_"
|arg1|arg2|...|argN}}
```

or for an explicit red error message

```
{{#invoke:check for unknown parameters|check
|unknown=<span class="error">Sorry, I don't recognize _VALUE_</span>
|arg1|arg2|...|argN}}
```

Here, `arg1`, `arg2`, ..., `argN`, are the known parameters. Unnamed (positional) parameters can be added too: `|1|2|argname1|argname2|...`. Any parameter which is used, but not on this list, will cause the module to return whatever is passed with the `unknown` parameter. The `_VALUE_` keyword, if used, will be changed to the name of the parameter. This is useful for either sorting the entries in a tracking category, or for provide more explicit information.

By default, the module makes no distinction between a defined-but-blank parameter and a non-blank parameter. That is, both unlisted Vorlage:Para and Vorlage:Para are reported. To only track non-blank parameters use Vorlage:Para.

By default, the module ignores blank positional parameters. That is, an unlisted Vorlage:Para is ignored. To *include* blank positional parameters in the tracking use Vorlage:Para.

Lua patterns

This module supports Lua patterns (similar to regular expressions), which are useful when there are many known parameters which use a systematic pattern. For example, template:infobox3cols uses

```
| regexp1 = header[%d][%d]*
| regexp2 = label[%d][%d]*
| regexp3 = data[%d][%d]*[abc]?
| regexp4 = class[%d][%d]*[abc]?
| regexp5 = rowclass[%d][%d]*
| regexp6 = rowstyle[%d][%d]*
| regexp7 = rowcellstyle[%d][%d]*
```

to match all parameters of the form `headerNUM`, `labelNUM`, `dataNUM`, `dataNUMa`, `dataNUMb`, `dataNUMc`, ..., `rowcellstyleNUM`, where NUM is a string of digits.

Example

```
{{Infobox
| above = {{{name|}}}

| label1 = Height
| data1 = {{{height|}}}

| label2 = Weight
| data2 = {{{weight|}}}

| label3 = Website
| data3 = {{{website|}}}
}}<!--
  end infobox, start tracking
-->{{#invoke:Check for unknown parameters|check
| unknown = {{main other|[[Category:Some tracking category|_VALUE_]]}}
| preview = unknown parameter "_VALUE_"
| name
| height | weight
| website
}}
```

See also

- Vorlage:Clc (category page can have header Vorlage:Tl)
- Module:Check for deprecated parameters – similar module that checks for deprecated parameters
- Module:Check for clobbered parameters – module that checks for conflicting parameters
- Module:TemplatePar – similar function (originally from dewiki)

- Template:Parameters and Module:Parameters – generates a list of parameter names for a given template
- Project:TemplateData based template parameter validation
- Module:Parameter validation checks a lot more
- User:Bamyers99/TemplateParametersTool - A tool for checking usage of template parameters

---

```
-- This module may be used to compare the arguments passed to the parent
-- with a list of arguments, returning a specified result if an argument is
-- not on the list
local p = {}

local function trim(s)
	return s:match('^%s*(.-)%s*$')
end

local function isnotempty(s)
	return s and s:match('%S')
end

local function clean(text)
	-- Return text cleaned for display and truncated if too long.
	-- Strip markers are replaced with dummy text representing the original w
	local pos, truncated
	local function truncate(text)
		if truncated then
			return ''
		end
		if mw.ustring.len(text) > 25 then
			truncated = true
			text = mw.ustring.sub(text, 1, 25) .. '...'
		end
		return mw.text.nowiki(text)
	end
	local parts = {}
	for before, tag, remainder in text:gmatch('([^\127]*)\127[^\127]*%-(%l+)%
		pos = remainder
		table.insert(parts, truncate(before) .. '&lt;' .. tag .. '&gt;...
	end
	table.insert(parts, truncate(text:sub(pos or 1)))
	return table.concat(parts)
end

function p._check(args, pargs)
	if type(args) ~= "table" or type(pargs) ~= "table" then
		-- TODO: error handling
		return
	end

	-- create the list of known args, regular expressions, and the return str
	local knownargs = {}
	local regexps = {}
	for k, v in pairs(args) do
		if type(k) == 'number' then
			v = trim(v)
			knownargs[v] = 1
		elseif k:find('^regexp[1-9][0-9]*$') then
			table.insert(regexps, '^' .. v .. '$')
		end
	end
```

```
	-- loop over the parent args, and make sure they are on the list
	local ignoreblank = isnotempty(args['ignoreblank'])
	local showblankpos = isnotempty(args['showblankpositional'])
	local values = {}
	for k, v in pairs(pargs) do
		if type(k) == 'string' and knownargs[k] == nil then
			local knownflag = false
			for _, regexp in ipairs(regexps) do
				if mw.ustring.match(k, regexp) then
					knownflag = true
					break
				end
			end
			if not knownflag and ( not ignoreblank or isnotempty(v) )
				table.insert(values, clean(k))
			end
		elseif type(k) == 'number' and knownargs[tostring(k)] == nil ther
			local knownflag = false
			for _, regexp in ipairs(regexps) do
				if mw.ustring.match(tostring(k), regexp) then
					knownflag = true
					break
				end
			end
			if not knownflag and ( showblankpos or isnotempty(v) ) th
				table.insert(values, k .. ' = ' .. clean(v))
			end
		end
	end

	-- add results to the output tables
	local res = {}
	if #values > 0 then
		local unknown_text = args['unknown'] or 'Found _VALUE_, '

		if mw.getCurrentFrame():preprocess( "{{REVISIONID}}" ) == "" ther
			local preview_text = args['preview']
			if isnotempty(preview_text) then
				preview_text = require('Module:If preview')._warn
			elseif preview == nil then
				preview_text = unknown_text
			end
			unknown_text = preview_text
		end
		for _, v in pairs(values) do
			-- Fix odd bug for | = which gets stripped to the empty s
			-- breaks category links
			if v == '' then v = ' ' end

			-- avoid error with v = 'example%2' ("invalid capture ind
			local r = unknown_text:gsub('_VALUE_', {_VALUE_ = v})
			table.insert(res, r)
		end
	end

	return table.concat(res)
end

function p.check(frame)
	local args = frame.args
	local pargs = frame:getParent().args
	return p._check(args, pargs)
end
```

```
return p
```

Modul:Check for unknown parameters/Doku

Dies ist die Dokumentationsseite für Modul:Check for unknown parameters

Vorlage:Lua

This module may be appended to a template to check for uses of unknown parameters.

Inhaltsverzeichnis

Usage

Basic usage

```
{{#invoke:check for unknown parameters|check
|unknown=[[Category:Some tracking category]]
|arg1|arg2|arg3|argN}}
```

or to sort the entries in the tracking category by parameter with a preview error message

```
{{#invoke:check for unknown parameters|check
|unknown=[[Category:Some tracking category|_VALUE_]]
|preview=unknown parameter "_VALUE_"
|arg1|arg2|...|argN}}
```

or for an explicit red error message

```
{{#invoke:check for unknown parameters|check
|unknown=<span class="error">Sorry, I don't recognize _VALUE_</span>
|arg1|arg2|...|argN}}
```

Here, `arg1`, `arg2`, ..., `argN`, are the known parameters. Unnamed (positional) parameters can be added too: `|1|2|argname1|argname2|...`. Any parameter which is used, but not on this list, will cause the module to return whatever is passed with the `unknown` parameter. The `_VALUE_` keyword, if used, will be changed to the name of the parameter. This is useful for either sorting the entries in a tracking category, or for provide more explicit information.

By default, the module makes no distinction between a defined-but-blank parameter and a non-blank parameter. That is, both unlisted Vorlage:Para and Vorlage:Para are reported. To only track non-blank parameters use Vorlage:Para.

By default, the module ignores blank positional parameters. That is, an unlisted Vorlage:Para is ignored. To *include* blank positional parameters in the tracking use Vorlage:Para.

Lua patterns

This module supports Lua patterns (similar to regular expressions), which are useful when there are many known parameters which use a systematic pattern. For example, template:infobox3cols uses

```
| regexp1 = header[%d][%d]*
| regexp2 = label[%d][%d]*
| regexp3 = data[%d][%d]*[abc]?
| regexp4 = class[%d][%d]*[abc]?
| regexp5 = rowclass[%d][%d]*
| regexp6 = rowstyle[%d][%d]*
| regexp7 = rowcellstyle[%d][%d]*
```

to match all parameters of the form `headerNUM`, `labelNUM`, `dataNUM`, `dataNUMa`, `dataNUMb`, `dataNUMc`, ..., `rowcellstyleNUM`, where NUM is a string of digits.

Example

```
{{Infobox
| above = {{{name|}}}

| label1 = Height
| data1 = {{{height|}}}

| label2 = Weight
| data2 = {{{weight|}}}

| label3 = Website
| data3 = {{{website|}}}
}}<!--
  end infobox, start tracking
-->{{#invoke:Check for unknown parameters|check
| unknown = {{main other|[[Category:Some tracking category|_VALUE_]]}}
| preview = unknown parameter "_VALUE_"
| name
| height | weight
| website
}}
```

See also

- Vorlage:Clc (category page can have header Vorlage:Tl)
- Module:Check for deprecated parameters – similar module that checks for deprecated parameters
- Module:Check for clobbered parameters – module that checks for conflicting parameters
- Module:TemplatePar – similar function (originally from dewiki)
- Template:Parameters and Module:Parameters – generates a list of parameter names for a given template

- Project:TemplateData based template parameter validation
- Module:Parameter validation checks a lot more
- User:Bamyers99/TemplateParametersTool - A tool for checking usage of template parameters