پودمان:Documentation

[illegible] [illegible]/Documentation:[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

```
-- برای فارسی‌سازی بهتر، بعضی از قسمت‌های این پودمان تغییر کرده‌است. لطفاً هنگام به‌روزرسانی به توضیحات پودمان توجه کنید.
-- This module implements {{documentation}}.

-- Get required modules.
local getArgs = require('Module:Arguments').getArgs

-- Get the config table.
local cfg = mw.loadData('Module:Documentation/config')

local p = {}

-- Often-used functions.
local ugsub = mw.ustring.gsub

---------------------------------------------------------------------------
-- Helper functions
--
-- These are defined as local functions, but are made available in the p
-- table for testing purposes.
---------------------------------------------------------------------------

local function message(cfgKey, valArray, expectType)
	--[[
	-- Gets a message from the cfg table and formats it if appropriate.
	-- The function raises an error if the value from the cfg table is
not
	-- of the type expectType. The default type for expectType is
'string'.
	-- If the table valArray is present, strings such as $1, $2 etc. in
the
	-- message are substituted with values from the table keys [1], [2]
etc.
	-- For example, if the message "foo-message" had the value 'Foo $2
bar $1.',
	-- message('foo-message', {'baz', 'qux'}) would return "Foo qux bar
baz."
	--]]
	local msg = cfg[cfgKey]
	expectType = expectType or 'string'
	if type(msg) ~= expectType then
		error('پیام: خطای نوع در متن cfg.' .. cfgKey .. ' (' ..
```

```
expectType .. ' انتظار می‌رفت، ' .. type(msg) .. ' 2 ,'(بدست آمد)
        end
        if not valArray then
                return msg
        end

        local function getMessageVal(match)
                match = tonumber(match)
                -- برای فارسی‌سازی بهتر، ساختار خطای نمایشی خط زیر تغییر
کرده‌است
                return valArray[match] or error('پیام: هیچ مقداری برای کلید
$' .. match .. ' در متن cfg.' .. cfgKey .. ' 4 ,'یافت نشد)
        end

        return ugsub(msg, '$([1-9۱-۹][0-9۰-۹]*)', getMessageVal)
end

p.message = message

local function makeWikilink(page, display)
        if display then
                return mw.ustring.format('[[%s|%s]]', page, display)
        else
                return mw.ustring.format('[[%s]]', page)
        end
end

p.makeWikilink = makeWikilink

local function makeCategoryLink(cat, sort)
        local catns = mw.site.namespaces[14].name
        return makeWikilink(catns .. ':' .. cat, sort)
end

p.makeCategoryLink = makeCategoryLink

local function makeUrlLink(url, display)
        return mw.ustring.format('[%s %s]', url, display)
end

p.makeUrlLink = makeUrlLink

local function makeToolbar(...)
        local ret = {}
        local lim = select('#', ...)
        if lim < 1 then
                return nil
        end
        for i = 1, lim do
                ret[#ret + 1] = select(i, ...)
        end
```

```
        -- 'documentation-toolbar'
        return '<span class="' .. message('toolbar-class') .. '">('
                .. table.concat(ret, ' &#124; ') .. ')</span>'
end

p.makeToolbar = makeToolbar

--------------------------------------------------------------------------------
-- Argument processing
--------------------------------------------------------------------------------

local function makeInvokeFunc(funcName)
        return function (frame)
                local args = getArgs(frame, {
                        valueFunc = function (key, value)
                                if type(value) == 'string' then
                                        value = mw.ustring.match(value,
'^%s*(.-)%s*$') -- Remove whitespace.
                                        if key == 'heading' or value ~= ''
then
                                                return value
                                        else
                                                return nil
                                        end
                                else
                                        return value
                                end
                        end
                })
                return p[funcName](args)
        end
end

--------------------------------------------------------------------------------
-- Entry points
--------------------------------------------------------------------------------

function p.nonexistent(frame)
        if mw.title.getCurrentTitle().subpageText == 'آزمایشی' then
                return frame:expandTemplate{title = 'هشدار صفحه آزمایشی
پودمان'}
        else
                return p.main(frame)
        end
end

p.main = makeInvokeFunc('_main')

function p._main(args)
        --[[
        -- This function defines logic flow for the module.
```

```
	-- @args - table of arguments passed by the user
	--]]
	local env = p.getEnvironment(args)
	local root = mw.html.create()
	root
		:wikitext(p._getModuleWikitext(args, env))
		:wikitext(p.protectionTemplate(env))
		:wikitext(p.sandboxNotice(args, env))
		:tag('div')
			-- 'documentation-container'
			:addClass(message('container'))
			:newline()
			:tag('div')
				-- 'documentation'
				:addClass(message('main-div-classes'))
				:newline()
				:wikitext(p._startBox(args, env))
				:wikitext(p._content(args, env))
				:tag('div')
					-- 'documentation-clear'
					:addClass(message('clear'))
					:done()
				:newline()
				:done()
			:wikitext(p._endBox(args, env))
			:done()
		:wikitext(p.addTrackingCategories(env))
	-- 'Module:Documentation/styles.css'
	return mw.getCurrentFrame():extensionTag (
		'templatestyles', '', {src=cfg['templatestyles']
	}) .. tostring(root)
end

--------------------------------------------------------------------------------
-- Environment settings
--------------------------------------------------------------------------------

function p.getEnvironment(args)
	--[[
	-- Returns a table with information about the environment, including
title
	-- objects and other namespace- or path-related data.
	-- @args - table of arguments passed by the user
	--
	-- Title objects include:
	-- env.title - the page we are making documentation for (usually the
current title)
	-- env.templateTitle - the template (or module, file, etc.)
	-- env.docTitle - the /doc subpage.
	-- env.sandboxTitle - the /sandbox subpage.
	-- env.testcasesTitle - the /testcases subpage.
```

```
	--
	-- Data includes:
	-- env.protectionLevels - the protection levels table of the title
object.
	-- env.subjectSpace - the number of the title's subject namespace.
	-- env.docSpace - the number of the namespace the title puts its
documentation in.
	-- env.docpageBase - the text of the base page of the /doc, /sandbox
and /testcases pages, with namespace.
	-- env.compareUrl - URL of the Special:ComparePages page comparing
the sandbox with the template.
	--
	-- All table lookups are passed through pcall so that errors are
caught. If an error occurs, the value
	-- returned will be nil.
	--]]
	local env, envFuncs = {}, {}

	-- Set up the metatable. If triggered we call the corresponding
function in the envFuncs table. The value
	-- returned by that function is memoized in the env table so that we
don't call any of the functions
	-- more than once. (Nils won't be memoized.)
	setmetatable(env, {
		__index = function (t, key)
			local envFunc = envFuncs[key]
			if envFunc then
				local success, val = pcall(envFunc)
				if success then
					env[key] = val -- Memoise the value.
					return val
				end
			end
			return nil
		end
	})

	function envFuncs.title()
		-- The title object for the current page, or a test page
passed with args.page.
		local title
		local titleArg = args.page
		if titleArg then
			title = mw.title.new(titleArg)
		else
			title = mw.title.getCurrentTitle()
		end
		return title
	end

	function envFuncs.templateTitle()
```

```
                --[[
                -- The template (or module, etc.) title object.
                -- Messages:
                -- 'sandbox-subpage' --> 'sandbox'
                -- 'testcases-subpage' --> 'testcases'
                --]]
                local subjectSpace = env.subjectSpace
                local title = env.title
                local subpage = title.subpageText
                if subpage == message('sandbox-subpage') or subpage ==
message('testcases-subpage') then
                        return mw.title.makeTitle(subjectSpace,
title.baseText)
                else
                        return mw.title.makeTitle(subjectSpace, title.text)
                end
        end

        function envFuncs.docTitle()
                --[[
                -- Title object of the /doc subpage.
                -- Messages:
                -- 'doc-subpage' --> 'doc'
                --]]
                local title = env.title
                local docname = args[1] -- User-specified doc page.
                local docpage
                if docname then
                        docpage = docname
                else
                        docpage = env.docpageBase .. '/' .. message('doc-
subpage')
                end
                return mw.title.new(docpage)
        end
        function envFuncs.sandboxTitle()
                --[[
                -- Title object for the /sandbox subpage.
                -- Messages:
                -- 'sandbox-subpage' --> 'sandbox'
                --]]
                return mw.title.new(env.docpageBase .. '/' ..
message('sandbox-subpage'))
        end
        function envFuncs.testcasesTitle()
                --[[
                -- Title object for the /testcases subpage.
                -- Messages:
                -- 'testcases-subpage' --> 'testcases'
                --]]
                return mw.title.new(env.docpageBase .. '/' ..
```

```
message('testcases-subpage'))
	end

	function envFuncs.protectionLevels()
		-- The protection levels table of the title object.
		return env.title.protectionLevels
	end

	function envFuncs.subjectSpace()
		-- The subject namespace number.
		return mw.site.namespaces[env.title.namespace].subject.id
	end

	function envFuncs.docSpace()
		-- The documentation namespace number. For most namespaces
this is the
		-- same as the subject namespace. However, pages in the
Article, File,
		-- MediaWiki or Category namespaces must have their /doc,
/sandbox and
		-- /testcases pages in talk space.
		local subjectSpace = env.subjectSpace
		if subjectSpace == 0 or subjectSpace == 6 or subjectSpace ==
8 or subjectSpace == 14 then
			return subjectSpace + 1
		else
			return subjectSpace
		end
	end

	function envFuncs.docpageBase()
		-- The base page of the /doc, /sandbox, and /testcases
subpages.
		-- For some namespaces this is the talk page, rather than the
template page.
		local templateTitle = env.templateTitle
		local docSpace = env.docSpace
		local docSpaceText = mw.site.namespaces[docSpace].name
		-- Assemble the link. docSpace is never the main namespace,
so we can hardcode the colon.
		return docSpaceText .. ':' .. templateTitle.text
	end
	function envFuncs.compareUrl()
		-- Diff link between the sandbox and the main template using
[[Special:ComparePages]].
		local templateTitle = env.templateTitle
		local sandboxTitle = env.sandboxTitle
		if templateTitle.exists and sandboxTitle.exists then
			local compareUrl = mw.uri.fullUrl(
				'ویژه:مقایسهٔ صفحات',
				{ page1 = templateTitle.prefixedText, page2 =
```

```
sandboxTitle.prefixedText}
                        )
                        return tostring(compareUrl)
                else
                        return nil
                end
        end

        return env
end

--------------------------------------------------------------------------------
-- Auxiliary templates
--------------------------------------------------------------------------------

p.getModuleWikitext = makeInvokeFunc('_getModuleWikitext')

function p._getModuleWikitext(args, env)
        local currentTitle = mw.title.getCurrentTitle()
        if currentTitle.contentModel ~= 'Scribunto' then return end
        pcall(require, currentTitle.prefixedText) -- if it fails, we don't
care
        local moduleWikitext =  package.loaded["Module:Module wikitext"]
        if moduleWikitext then
                return moduleWikitext.main()
        end
end

function p.sandboxNotice(args, env)
        --[=[
        -- Generates a sandbox notice for display above sandbox pages.
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated
with p.getEnvironment
        --
        -- Messages:
        -- 'sandbox-notice-image' --> '[[Image:Sandbox.svg|50px|alt=|link=]]'
        -- 'sandbox-notice-blurb' --> 'This is the $1 for $2.'
        -- 'sandbox-notice-diff-blurb' --> 'This is the $1 for $2 ($3).'
        -- 'sandbox-notice-pagetype-template' --> '[[Wikipedia:Template test
cases|template sandbox]] page'
        -- 'sandbox-notice-pagetype-module' --> '[[Wikipedia:Template test
cases|module sandbox]] page'
        -- 'sandbox-notice-pagetype-other' --> 'sandbox page'
        -- 'sandbox-notice-compare-link-display' --> 'diff'
        -- 'sandbox-notice-testcases-blurb' --> 'See also the companion
subpage for $1.'
        -- 'sandbox-notice-testcases-link-display' --> 'test cases'
        -- 'sandbox-category' --> 'Template sandboxes'
        --]=]
        local title = env.title
```

```
        local sandboxTitle = env.sandboxTitle
        local templateTitle = env.templateTitle
        local subjectSpace = env.subjectSpace
        if not (subjectSpace and title and sandboxTitle and templateTitle
                and mw.title.equals(title, sandboxTitle)) then
                return nil
        end
        -- Build the table of arguments to pass to {{ombox}}. We need just
two fields, "image" and "text".
        local omargs = {}
        omargs.image = message('sandbox-notice-image')
        -- Get the text. We start with the opening blurb, which is something
like
        -- "This is the template sandbox for [[Template:Foo]] (diff)."
        local text = ''
        local pagetype
        if subjectSpace == 10 then
                pagetype = message('sandbox-notice-pagetype-template')
        elseif subjectSpace == 828 then
                pagetype = message('sandbox-notice-pagetype-module')
        else
                pagetype = message('sandbox-notice-pagetype-other')
        end
        local templateLink = makeWikilink(templateTitle.prefixedText)
        local compareUrl = env.compareUrl
        if compareUrl then
                local compareDisplay = message('sandbox-notice-compare-link-
display')
                local compareLink = makeUrlLink(compareUrl, compareDisplay)
                text = text .. message('sandbox-notice-diff-blurb',
{pagetype, templateLink, compareLink})
        else
                text = text .. message('sandbox-notice-blurb', {pagetype,
templateLink})
        end
        -- Get the test cases page blurb if the page exists. This is
something like
        -- "See also the companion subpage for [[Template:Foo/testcases|test
cases]]."
        local testcasesTitle = env.testcasesTitle
        if testcasesTitle and testcasesTitle.exists then
                if testcasesTitle.contentModel == "Scribunto" then
                        local testcasesLinkDisplay = message('sandbox-notice-
testcases-link-display')
                        local testcasesRunLinkDisplay = message('sandbox-
notice-testcases-run-link-display')
                        local testcasesLink =
makeWikilink(testcasesTitle.prefixedText, testcasesLinkDisplay)
                        local testcasesRunLink =
makeWikilink(testcasesTitle.talkPageTitle.prefixedText,
testcasesRunLinkDisplay)
```

```
                        text = text .. '<br />' .. message('sandbox-notice-
testcases-run-blurb', {testcasesLink, testcasesRunLink})
                else
                        local testcasesLinkDisplay = message('sandbox-notice-
testcases-link-display')
                        local testcasesLink =
makeWikilink(testcasesTitle.prefixedText, testcasesLinkDisplay)
                        text = text .. '<br />' .. message('sandbox-notice-
testcases-blurb', {testcasesLink})
                end
        end
        -- Add the sandbox to the sandbox category.
        omargs.text = text .. makeCategoryLink(message('sandbox-category'))

        -- 'documentation-clear'
        return '<div class="' .. message('clear') .. '"></div>'
                .. require('Module:Message box').main('ombox', omargs)
end

function p.protectionTemplate(env)
        -- Generates the padlock icon in the top right.
        -- @env - environment table containing title objects, etc., generated
with p.getEnvironment
        -- Messages:
        -- 'protection-template' --> 'pp-template'
        -- 'protection-template-args' --> {docusage = 'yes'}
        local protectionLevels = env.protectionLevels
        if not protectionLevels then
                return nil
        end
        local editProt = protectionLevels.edit and protectionLevels.edit[1]
        local moveProt = protectionLevels.move and protectionLevels.move[1]
        if editProt then
                -- The page is edit-protected.
                return require('Module:Protection banner')._main{
                        message('protection-reason-edit'), small = true
                }
        elseif moveProt and moveProt ~= 'autoconfirmed' then
                -- The page is move-protected but not edit-protected. Exclude
move
                -- protection with the level "autoconfirmed", as this is
equivalent to
                -- no move protection at all.
                return require('Module:Protection banner')._main{
                        action = 'move', small = true
                }
        else
                return nil
        end
end
```

```
--------------------------------------------------------------------------------
-- Start box
--------------------------------------------------------------------------------

p.startBox = makeInvokeFunc('_startBox')

function p._startBox(args, env)
	--[[
	-- This function generates the start box.
	-- @args - a table of arguments passed by the user
	-- @env - environment table containing title objects, etc., generated
with p.getEnvironment
	--
	-- The actual work is done by p.makeStartBoxLinksData and
p.renderStartBoxLinks which make
	-- the [view] [edit] [history] [purge] links, and by
p.makeStartBoxData and p.renderStartBox
	-- which generate the box HTML.
	--]]
	env = env or p.getEnvironment(args)
	local links
	local content = args.content or args['محتوا']
	if not content or args[1] then
		-- No need to include the links if the documentation is on
the template page itself.
		local linksData = p.makeStartBoxLinksData(args, env)
		if linksData then
			links = p.renderStartBoxLinks(linksData)
		end
	end
	-- Generate the start box html.
	local data = p.makeStartBoxData(args, env, links)
	if data then
		return p.renderStartBox(data)
	else
		-- User specified no heading.
		return nil
	end
end

function p.makeStartBoxLinksData(args, env)
	--[[
	-- Does initial processing of data to make the [view] [edit]
[history] [purge] links.
	-- @args - a table of arguments passed by the user
	-- @env - environment table containing title objects, etc., generated
with p.getEnvironment
	--
	-- Messages:
	-- 'view-link-display' --> 'view'
	-- 'edit-link-display' --> 'edit'
```

```
        -- 'history-link-display' --> 'history'
        -- 'purge-link-display' --> 'purge'
        -- 'file-docpage-preload' --> 'Template:Documentation/preload-
filespace'
        -- 'module-preload' --> 'Template:Documentation/preload-module-doc'
        -- 'docpage-preload' --> 'Template:Documentation/preload'
        -- 'create-link-display' --> 'create'
        --]]
        local subjectSpace = env.subjectSpace
        local title = env.title
        local docTitle = env.docTitle
        if not title or not docTitle then
                return nil
        end
        if docTitle.isRedirect then
                docTitle = docTitle.redirectTarget
        end

        local data = {}
        data.title = title
        data.docTitle = docTitle
        -- View, display, edit, and purge links if /doc exists.
        data.viewLinkDisplay = message('view-link-display')
        data.editLinkDisplay = message('edit-link-display')
        data.historyLinkDisplay = message('history-link-display')
        data.purgeLinkDisplay = message('purge-link-display')
        -- Create link if /doc doesn't exist.
        local preload = args.preload
        if not preload then
                if subjectSpace == 6 then -- File namespace
                        preload = message('file-docpage-preload')
                elseif subjectSpace == 828 then -- Module namespace
                        preload = message('module-preload')
                else
                        preload = message('docpage-preload')
                end
        end
        data.preload = preload
        data.createLinkDisplay = message('create-link-display')
        return data
end

function p.renderStartBoxLinks(data)
        --[[
        -- Generates the [view][edit][history][purge] or [create][purge]
links from the data table.
        -- @data - a table of data generated by p.makeStartBoxLinksData
        --]]
        local function escapeBrackets(s)
                -- Escapes square brackets with HTML entities.
                s = s:gsub('%[', '&#91;') -- Replace square brackets with
```

```
HTML entities.
                s = s:gsub('%]', '&#93;')
                return s
        end

        local ret
        local docTitle = data.docTitle
        local title = data.title
        local purgeLink = makeUrlLink(title:fullUrl{action = 'purge'},
data.purgeLinkDisplay)
        if docTitle.exists then
                local viewLink = makeWikilink(docTitle.prefixedText,
data.viewLinkDisplay)
                local editLink = makeUrlLink(docTitle:fullUrl{action =
'edit'}, data.editLinkDisplay)
                local historyLink = makeUrlLink(docTitle:fullUrl{action =
'history'}, data.historyLinkDisplay)
                ret = '[%s] [%s] [%s] [%s]'
                ret = escapeBrackets(ret)
                ret = mw.ustring.format(ret, viewLink, editLink, historyLink,
purgeLink)
        else
                local createLink = makeUrlLink(docTitle:fullUrl{action =
'edit', preload = data.preload}, data.createLinkDisplay)
                ret = '[%s] [%s]'
                ret = escapeBrackets(ret)
                ret = mw.ustring.format(ret, createLink, purgeLink)
        end
        return ret
end

function p.makeStartBoxData(args, env, links)
        --[=[
        -- Does initial processing of data to pass to the start-box render
function, p.renderStartBox.
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated
with p.getEnvironment
        -- @links - a string containing the [view][edit][history][purge]
links - could be nil if there's an error.
        --
        -- Messages:
        -- 'documentation-icon-wikitext' --> '[[File:Test Template Info-Icon
- Version (2).svg|50px|link=|alt=]]'
        -- 'template-namespace-heading' --> 'Template documentation'
        -- 'module-namespace-heading' --> 'Module documentation'
        -- 'file-namespace-heading' --> 'Summary'
        -- 'other-namespaces-heading' --> 'Documentation'
        -- 'testcases-create-link-display' --> 'create'
        --]=]
        local subjectSpace = env.subjectSpace
```

```
        if not subjectSpace then
                -- Default to an "other namespaces" namespace, so that we get
at least some output
                -- if an error occurs.
                subjectSpace = 2
        end
        local data = {}
        -- Heading
        local heading = args.heading -- Blank values are not removed.
        if heading == '' then
                -- Don't display the start box if the heading arg is defined
but blank.
                return nil
        end
        if heading then
                data.heading = heading
        elseif subjectSpace == 10 then -- Template namespace
                data.heading = message('documentation-icon-wikitext') .. ' '
.. message('template-namespace-heading')
        elseif subjectSpace == 828 then -- Module namespace
                data.heading = message('documentation-icon-wikitext') .. ' '
.. message('module-namespace-heading')
        elseif subjectSpace == 6 then -- File namespace
                data.heading = message('file-namespace-heading')
        else
                data.heading = message('other-namespaces-heading')
        end
        -- Heading CSS
        local headingStyle = args['heading-style']
        if headingStyle then
                data.headingStyleText = headingStyle
        else
                -- 'documentation-heading'
                data.headingClass = message('main-div-heading-class')
        end
        -- Data for the [view][edit][history][purge] or [create] links.
        if links then
                -- 'mw-editsection-like plainlinks'
                data.linksClass = message('start-box-link-classes')
                data.links = links
        end
        return data
end

function p.renderStartBox(data)
        -- Renders the start box html.
        -- @data - a table of data generated by p.makeStartBoxData.
        local sbox = mw.html.create('div')
        sbox
                -- 'documentation-startbox'
                :addClass(message('start-box-class'))
```

```
                :newline()
                :tag('span')
                        :addClass(data.headingClass)
                        :cssText(data.headingStyleText)
                        :wikitext(data.heading)
        local links = data.links
        if links then
                sbox:tag('span')
                        :addClass(data.linksClass)
                        :attr('id', data.linksId)
                        :wikitext(links)
        end
        return tostring(sbox)
end

--------------------------------------------------------------------------------
-- Documentation content
--------------------------------------------------------------------------------

p.content = makeInvokeFunc('_content')

function p._content(args, env)
        -- Displays the documentation contents
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated
with p.getEnvironment
        env = env or p.getEnvironment(args)
        local docTitle = env.docTitle
        local content = args.content or args['محتوا']
        if not content and docTitle and docTitle.exists then
                content = args._content or
mw.getCurrentFrame():expandTemplate{title = docTitle.prefixedText}
        end
        -- The line breaks below are necessary so that "=== Headings ===" at
the start and end
        -- of docs are interpreted correctly.
        return '\n' .. (content or '') .. '\n'
end

p.contentTitle = makeInvokeFunc('_contentTitle')

function p._contentTitle(args, env)
        env = env or p.getEnvironment(args)
        local docTitle = env.docTitle
        if not (args.content or args['محتوا']) and docTitle and
docTitle.exists then
                return docTitle.prefixedText
        else
                return ''
        end
end
```

```
--------------------------------------------------------------------------------
-- End box
--------------------------------------------------------------------------------

p.endBox = makeInvokeFunc('_endBox')

function p._endBox(args, env)
	--[=[
	-- This function generates the end box (also known as the link box).
	-- @args - a table of arguments passed by the user
	-- @env - environment table containing title objects, etc., generated
with p.getEnvironment
	--
	--]=]
	-- Get environment data.
	env = env or p.getEnvironment(args)
	local subjectSpace = env.subjectSpace
	local docTitle = env.docTitle
	if not subjectSpace or not docTitle then
		return nil
	end
	-- Check whether we should output the end box at all. Add the end
	-- box by default if the documentation exists or if we are in the
	-- user, module or template namespaces.
	local linkBox = args['link box']
	if linkBox == 'off'
		or not (
			docTitle.exists
			or subjectSpace == 2
			or subjectSpace == 828
			or subjectSpace == 10
		)
	then
		return nil
	end

	-- Assemble the link box.
	local text = ''
	if linkBox then
		text = text .. linkBox
	else
		text = text .. (p.makeDocPageBlurb(args, env) or '') -- "This
documentation is transcluded from [[Foo]]."
		if subjectSpace == 2 or subjectSpace == 10 or subjectSpace ==
828 then
			-- We are in the user, template or module namespaces.
			-- Add sandbox and testcases links.
			-- "Editors can experiment in this template's sandbox
and testcases pages."
			text = text .. (p.makeExperimentBlurb(args, env) or
'') .. '<br />'
```

```
                        if not (args.content or args['محتوا']) and not
args[1] then
                                -- "Please add categories to the /doc
subpage."
                                -- Don't show this message with inline docs
or with an explicitly specified doc page,
                                -- as then it is unclear where to add the
categories.
                                text = text .. (p.makeCategoriesBlurb(args,
env) or '')
                        end
                        text = text .. ' ' .. (p.makeSubpagesBlurb(args, env)
or '') --"Subpages of this template"
                end
        end
        local box = mw.html.create('div')
        -- 'documentation-metadata'
        box:attr('role', 'note')
                :addClass(message('end-box-class'))
                -- 'plainlinks'
                :addClass(message('end-box-plainlinks'))
                :wikitext(text)
                :done()

        return '\n' .. tostring(box)
end

function p.makeDocPageBlurb(args, env)
        --[=[
        -- Makes the blurb "This documentation is transcluded from
[[Template:Foo]] (edit, history)".
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated
with p.getEnvironment
        --
        -- Messages:
        -- 'edit-link-display' --> 'edit'
        -- 'history-link-display' --> 'history'
        -- 'transcluded-from-blurb' -->
        -- 'The above [[Wikipedia:Template documentation|documentation]]
        -- is [[Help:Transclusion|transcluded]] from $1.'
        -- 'module-preload' --> 'Template:Documentation/preload-module-doc'
        -- 'create-link-display' --> 'create'
        -- 'create-module-doc-blurb' -->
        -- 'You might want to $1 a documentation page for this
[[Wikipedia:Lua|Scribunto module]].'
        --]=]
        local docTitle = env.docTitle
        if not docTitle then
                return nil
        end
```

```
        local ret
        if docTitle.exists then
                -- /doc exists; link to it.
                local docLink = makeWikilink(docTitle.prefixedText)
                local editUrl = docTitle:fullUrl{action = 'edit'}
                local editDisplay = message('edit-link-display')
                local editLink = makeUrlLink(editUrl, editDisplay)
                local historyUrl = docTitle:fullUrl{action = 'history'}
                local historyDisplay = message('history-link-display')
                local historyLink = makeUrlLink(historyUrl, historyDisplay)
                ret = message('transcluded-from-blurb', {docLink})
                        .. ' '
                        .. makeToolbar(editLink, historyLink)
                        .. '<br />'
        elseif env.subjectSpace == 828 then
                -- /doc does not exist; ask to create it.
                local createUrl = docTitle:fullUrl{action = 'edit', preload =
message('module-preload')}
                local createDisplay = message('create-link-display')
                local createLink = makeUrlLink(createUrl, createDisplay)
                ret = message('create-module-doc-blurb', {createLink})
                        .. '<br />'
        end
        return ret
end

function p.makeExperimentBlurb(args, env)
        --[[
        -- Renders the text "Editors can experiment in this template's
sandbox (edit | diff) and testcases (edit) pages."
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated
with p.getEnvironment
        --
        -- Messages:
        -- 'sandbox-link-display' --> 'sandbox'
        -- 'sandbox-edit-link-display' --> 'edit'
        -- 'compare-link-display' --> 'diff'
        -- 'module-sandbox-preload' --> 'Template:Documentation/preload-
module-sandbox'
        -- 'template-sandbox-preload' --> 'Template:Documentation/preload-
sandbox'
        -- 'sandbox-create-link-display' --> 'create'
        -- 'mirror-edit-summary' --> 'Create sandbox version of $1'
        -- 'mirror-link-display' --> 'mirror'
        -- 'mirror-link-preload' --> 'Template:Documentation/mirror'
        -- 'sandbox-link-display' --> 'sandbox'
        -- 'testcases-link-display' --> 'testcases'
        -- 'testcases-edit-link-display'--> 'edit'
        -- 'template-sandbox-preload' --> 'Template:Documentation/preload-
sandbox'
```

```
        -- 'testcases-create-link-display' --> 'create'
        -- 'testcases-link-display' --> 'testcases'
        -- 'testcases-edit-link-display' --> 'edit'
        -- 'module-testcases-preload' --> 'Template:Documentation/preload-
module-testcases'
        -- 'template-testcases-preload' --> 'Template:Documentation/preload-
testcases'
        -- 'experiment-blurb-module' --> 'Editors can experiment in this
module's $1 and $2 pages.'
        -- 'experiment-blurb-template' --> 'Editors can experiment in this
template's $1 and $2 pages.'
        --]]
        local subjectSpace = env.subjectSpace
        local templateTitle = env.templateTitle
        local sandboxTitle = env.sandboxTitle
        local testcasesTitle = env.testcasesTitle
        local templatePage = templateTitle.prefixedText
        if not subjectSpace or not templateTitle or not sandboxTitle or not
testcasesTitle then
                return nil
        end
        -- Make links.
        local sandboxLinks, testcasesLinks
        if sandboxTitle.exists then
                local sandboxPage = sandboxTitle.prefixedText
                local sandboxDisplay = message('sandbox-link-display')
                local sandboxLink = makeWikilink(sandboxPage, sandboxDisplay)
                local sandboxEditUrl = sandboxTitle:fullUrl{action = 'edit'}
                local sandboxEditDisplay = message('sandbox-edit-link-
display')
                local sandboxEditLink = makeUrlLink(sandboxEditUrl,
sandboxEditDisplay)
                local compareUrl = env.compareUrl
                local compareLink
                if compareUrl then
                        local compareDisplay = message('compare-link-
display')
                        compareLink = makeUrlLink(compareUrl, compareDisplay)
                end
                sandboxLinks = sandboxLink .. ' ' ..
makeToolbar(sandboxEditLink, compareLink)
        else
                local sandboxPreload
                if subjectSpace == 828 then
                        sandboxPreload = message('module-sandbox-preload')
                else
                        sandboxPreload = message('template-sandbox-preload')
                end
                local sandboxCreateUrl = sandboxTitle:fullUrl{action =
'edit', preload = sandboxPreload}
                local sandboxCreateDisplay = message('sandbox-create-link-
```

```
display')
                local sandboxCreateLink = makeUrlLink(sandboxCreateUrl,
sandboxCreateDisplay)
                local mirrorSummary = message('mirror-edit-summary',
{makeWikilink(templatePage)})
                local mirrorPreload = message('mirror-link-preload')
                local mirrorUrl = sandboxTitle:fullUrl{action = 'edit',
preload = mirrorPreload, summary = mirrorSummary}
                if subjectSpace == 828 then
                        mirrorUrl = sandboxTitle:fullUrl{action = 'edit',
preload = templateTitle.prefixedText, summary = mirrorSummary}
                end
                local mirrorDisplay = message('mirror-link-display')
                local mirrorLink = makeUrlLink(mirrorUrl, mirrorDisplay)
                sandboxLinks = message('sandbox-link-display') .. ' ' ..
makeToolbar(sandboxCreateLink, mirrorLink)
        end
        if testcasesTitle.exists then
                local testcasesPage = testcasesTitle.prefixedText
                local testcasesDisplay = message('testcases-link-display')
                local testcasesLink = makeWikilink(testcasesPage,
testcasesDisplay)
                local testcasesEditUrl = testcasesTitle:fullUrl{action =
'edit'}
                local testcasesEditDisplay = message('testcases-edit-link-
display')
                local testcasesEditLink = makeUrlLink(testcasesEditUrl,
testcasesEditDisplay)
                -- for Modules, add testcases run link if exists
                if testcasesTitle.contentModel == "Scribunto"  and
testcasesTitle.talkPageTitle and testcasesTitle.talkPageTitle.exists then
                        local testcasesRunLinkDisplay = message('testcases-
run-link-display')
                        local testcasesRunLink =
makeWikilink(testcasesTitle.talkPageTitle.prefixedText,
testcasesRunLinkDisplay)
                        testcasesLinks = testcasesLink .. ' ' ..
makeToolbar(testcasesEditLink, testcasesRunLink)
                else
                        testcasesLinks = testcasesLink .. ' ' ..
makeToolbar(testcasesEditLink)
                end
        else
                local testcasesPreload
                if subjectSpace == 828 then
                        testcasesPreload = message('module-testcases-
preload')
                else
                        testcasesPreload = message('template-testcases-
preload')
                end
```

```
                local testcasesCreateUrl = testcasesTitle:fullUrl{action =
'edit', preload = testcasesPreload}
                local testcasesCreateDisplay = message('testcases-create-
link-display')
                local testcasesCreateLink = makeUrlLink(testcasesCreateUrl,
testcasesCreateDisplay)
                testcasesLinks = message('testcases-link-display') .. ' ' ..
makeToolbar(testcasesCreateLink)
        end
        local messageName
        if subjectSpace == 828 then
                messageName = 'experiment-blurb-module'
        else
                messageName = 'experiment-blurb-template'
        end
        return message(messageName, {sandboxLinks, testcasesLinks})
end

function p.makeCategoriesBlurb(args, env)
        --[[
        -- Generates the text "Please add categories to the /doc subpage."
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated
with p.getEnvironment
        -- Messages:
        -- 'doc-link-display' --> '/doc'
        -- 'add-categories-blurb' --> 'Please add categories to the $1
subpage.'
        --]]
        local docTitle = env.docTitle
        if not docTitle then
                return nil
        end
        local docPathLink = makeWikilink(docTitle.prefixedText, message('doc-
link-display'))
        return message('add-categories-blurb', {docPathLink})
end

function p.makeSubpagesBlurb(args, env)
        --[[
        -- Generates the "Subpages of this template" link.
        -- @args - a table of arguments passed by the user
        -- @env - environment table containing title objects, etc., generated
with p.getEnvironment
        -- Messages:
        -- 'template-pagetype' --> 'template'
        -- 'module-pagetype' --> 'module'
        -- 'default-pagetype' --> 'page'
        -- 'subpages-link-display' --> 'Subpages of this $1'
        --]]
        local subjectSpace = env.subjectSpace
```

```
        local templateTitle = env.templateTitle
        if not subjectSpace or not templateTitle then
                return nil
        end
        local pagetype
        if subjectSpace == 10 then
                pagetype = message('template-pagetype')
        elseif subjectSpace == 828 then
                pagetype = message('module-pagetype')
        else
                pagetype = message('default-pagetype')
        end
        local subpagesLink = makeWikilink(
                'ویژه:نمایه پیشوندی/' .. templateTitle.prefixedText .. '/',
                message('subpages-link-display', {pagetype})
        )
        return message('subpages-blurb', {subpagesLink})
end

--------------------------------------------------------------------------------
-- Tracking categories
--------------------------------------------------------------------------------

function p.addTrackingCategories(env)
        --[[
        -- Check if {{documentation}} is transcluded on a /doc or /testcases
page.
        -- @env - environment table containing title objects, etc., generated
with p.getEnvironment
        -- Messages:
        -- 'display-strange-usage-category' --> true
        -- 'doc-subpage' --> 'doc'
        -- 'testcases-subpage' --> 'testcases'
        -- 'strange-usage-category' --> 'Wikipedia pages with strange
((documentation)) usage'
        --
        -- /testcases pages in the module namespace are not categorised, as
they may have
        -- {{documentation}} transcluded automatically.
        --]]
        local title = env.title
        local subjectSpace = env.subjectSpace
        if not title or not subjectSpace then
                return nil
        end
        local subpage = title.subpageText
        local ret = ''
        if message('display-strange-usage-category', nil, 'boolean')
                and (
                        subpage == message('doc-subpage')
                        or subjectSpace ~= 828 and subpage ==
```

```
message('testcases-subpage')
                )
        then
                ret = ret .. makeCategoryLink(message('strange-usage-
category'))
        end
        return ret
end

return p
```