



Modul:Convert/tester

This module runs unit tests to compare template output with expected text. In addition, the module can output the results of expanding templates.

While intended for testing [Module:Convert](#), the tester should be useful with other templates that require many tests using a simple format for the test input.

Inhaltsverzeichnis	
1 Testcases example	1
2 Format	2
3 Specifying tests	2
4 Running tests from any page	3
5 Making expected results	3
6 Using show=all	4
7 Comparing a module with its sandbox	4

Testcases example

- [Module:Convert/sandbox/testcases](#) • templates to be tested, with expected outputs
- [Module talk:Convert/sandbox/testcases](#) • view test results

It is not necessary to save the testcases page before viewing test results. For example, [Module:Convert/sandbox/testcases](#) could be edited to change the tests. While still editing that page, paste [Vorlage:Nowrap](#) (without quotes) into the page title box under "Preview page with this template", then click "Show preview".

The testcases talk page (for example, [Module talk:Convert/sandbox/testcases](#)) contains:

```
{#invoke:convert/sandbox/testcases|run_tests}}
```

The testcases module page (for example, [Module:Convert/sandbox/testcases](#)) may contain:

```
local tests = [=[
A template to be tested must be at the start of a line.
Lines which do not start with a template are ignored.
{{convert/sandbox|1|acre|lk=on}}          1 [[acre]] (0.40&nbsp;[[hectare|ha]])
{{convert/sandbox|1|m2|acres|lk=on}}    1 [[square metre]] (0.00025 [[acre]]s)
{{convert/sandbox|0.16|/|2|disp=table}}  align="right"|0.16\n|align="right"|0.16

]=]=]

local p = require('Module:Convert/tester')
p.tests = tests
return p
```



If wanted, the tests can be run using a template different from the one specified in the tests. For example, the following would run the tests from [Module:Convert/sandbox/testcases](#), but would change the name of each template found on that page to "convert/sandbox2".

```
{#{invoke:convert/sandbox/testcases|run_tests|template=convert/sandbox2}}
```

Format

Tests are extracted from a multiline string. Any line that does not start with a template is ignored. Each processed line starts with a template, and is followed by whitespace, then the wikitext which should result from expanding the template.

The expected output must be entered in a single line. If the template outputs multiple lines, those lines must be joined with "\n" (two characters—backslash n).

The templates do not have to be the same, for example, the following tests would work:

```
local tests = [=[
{{convert|12|m}}                12 metres (39&nbsp;ft)
{{convert/sandbox|12|m}}       12 metres (39&nbsp;ft)
{{age|1989|7|23|2003|7|14}}    13
{{age in days|2007|5|24|2008|4|23}} 335
]=]
```

In the results, the status column shows "Pass" if the output from the template exactly matches the expected text. If there is no expected text, the template output is shown in the Actual column with a blank status. If the given expected text differs from the template output, the template output is shown in the Actual column with status "Fail", and the number of fails is shown at the top of the page. Searching the page for "Fail" will find each problem. Any "Fail" result is followed by a row showing the nowiki actual and expected wikitext.

Specifying tests

If using a testcases module (as in the [above example](#)), the test text is assigned to p. tests before executing run_tests.

Alternatively, the test text can be read from any page, or from any section on any page. For example, the following wikitext could be entered in a sandbox:

```
== Mixed tests ==
<pre>
{{convert|12|m}}                12 metres (39 ft)
{{convert/sandbox|0.16|/l|2|disp=table}} align="right"|0.16\n|align="right"|0
{{age in days|2007|5|24|2008|4|23}} 335
--- The following line is incorrect to demonstrate a "fail".
{{convert|12|m|lk=on}}          12 [[meter|metres]] (39 [[Foot|ft]])
The following line demonstrates the result when no expected text is provided.
{{convert/sandbox|1|-|5|in|mm|lk=on}}
</pre>
```

Given the above, the tests can be run as shown in the [following section](#).

Instead of specifying the tests with a multiline string, it is possible to assign a table to `p.tests` as shown in the following testcases module.

```
local tests = {
  -- Each test item is of form { template, expected }.
  { '{{convert|12|m}}', '12 metres (39 ft)' },
  { '{{convert/sandbox|0.16|/l|2|disp=table}}', 'align="right"|0.16\n|align="r' },
  { '{{age in days|2007|5|24|2008|4|23}}', '335' },
  { '{{convert|12|m|lk=on}}', '12 [[meter|metres]] (39 [[Foot|ft]])' },
  { '{{convert/sandbox|1|-|5|in|mm|lk=on}}' },
}

local p = require('Module:Convert/tester')
p.tests = tests
return p
```

This example provides the same results as the multiline string at "Mixed tests" above.

Running tests from any page

Entering either of the following lines of wikitext in a sandbox or talk page would run the tests found at the specified location. The first line would show all tests on page "Template talk: Example", while the second would show only those tests on that page that are in the "Mixed tests" section.

```
{{#invoke:convert/tester|run_tests|page=Template talk:Example}}
{{#invoke:convert/tester|run_tests|page=Template talk:Example|section=Mixed tests}}
```

As a demonstration, the following line is used to produce the table shown below, including the comment that starts with three dashes.

```
{{#invoke:convert/tester|run_tests|page=Module:Convert/tester/doc|section=Specify}}
```

Error

Could not read wikitext from "**Module:Convert/tester/doc**".

Making expected results

Function `make_tests` can be used to create tests in the format expected by `run_tests`. For example, previewing either of the following in a sandbox would show the results from expanding each template found on the specified page.

```
{{#invoke:convert/tester|make_tests|page=Template talk:Example}}
{{#invoke:convert/tester|make_tests|page=Template talk:Example|show=all}}
```



When using `make_tests`, any expected results in the input are ignored. Instead, the module shows each template and its actual output as plain text which can be copied to make a testcases page. The templates to be processed can be specified by setting `p.tests` or by specifying a page with an optional section.

If `|show=all` is included, any non-template lines are included in the result. The output could then be copied and used to replace the page with the tests in order to update the expected text for each template, but without changing non-template lines.

As a demonstration, the following line is used to produce the text shown below.

```
{{#invoke:convert/tester|make_tests|page=Module:Convert/tester/doc|section=Specif
```

Error

Could not read wikitext from "**Module:Convert/tester/doc**".

Using `show=all`

The `|show=all` option can be used with `make_tests` and with `run_tests`.

An example using `make_tests` is shown in the [previous section](#).

Using `|show=all` with `run_tests` allows comment lines to be displayed in the output table—not `//` lines are shown, only those that start with three dashes. For example, the testcases may include the following.

```
Added 12 January 2014.  
--- The following tests check the widget option.  
{{example|1|2|widget=on}}          ...(expected output)...
```

The table produced by `run_tests` would show "The following tests check the widget option." as a comment line, but only if `|show=all` is used. Comments have a distinctive background color, but also show "Cmnt" in the status column so they can be found by searching.

Comparing a module with its sandbox

When viewing a module, the documentation page is displayed; if the module has a sandbox, the documentation includes "Editors can experiment in this module's sandbox" with a link to [diff](#) the module and its sandbox.

The tester module provides a `compare` function which can check a series of modules, and compare each with its sandbox. A table is displayed showing whether the content is different, with a [diff](#) link.

For example, the following wikitext could be used.

```
{{#invoke:convert/tester|compare|Example|Example/data}}
```



The names "Example" and "Example/data" do not include a colon (:), so "Module:" is assumed. The command compares **Module:Example** with **Module:Example/sandbox**, and **Module:Example/data** with **Module:Example/data/sandbox**.

It is also possible for a module to define pairs of page titles in `p.pairs` (a table), and to use the tester module to generate a table for each pair of titles.

As a convenience, certain keywords are defined. If a keyword is recognized, the list of pairs comes from the module rather than the parameters. For example, the following uses the "convert" keyword to get the list of pairs of pages related to Module:Convert.

```
{#{#invoke:convert/tester|compare|convert}}
```

The following text is a sample showing output that may result from the above.

- `Module:Convert` • `Module:Convert/sandbox` • `different (diff)`
- `Module:Convert/data` • `Module:Convert/data/sandbox` • `same content`
- `Module:Convert/text` • `Module:Convert/text/sandbox` • `different (diff)`
- `Module:Convert/extra` • `Module:Convert/extra/sandbox` • `different (diff)`

By default, each output line is prefixed with '*' to give a bulleted list. An alternative prefix can be specified with the prefix parameter. For example, the following gives an indented bulleted list.

```
{#{#invoke:convert/tester|compare|convert|prefix=*}}
```

```
-- Test the output from a template by comparing it with fixed text.
-- The expected text must be in a single line, but can include
-- "\n" (two characters) to indicate that a newline is expected.
-- Tests are run (or created) by setting p.tests (string or table), or
-- by setting page=PAGE_TITLE (and optionally section=SECTION_TITLE),
-- then executing run_tests (or make_tests).
```

```
local Collection = {}
Collection.__index = Collection
do
    function Collection:add(item)
        if item ~= nil then
            self.n = self.n + 1
            self[self.n] = item
        end
    end
    function Collection:join(sep)
        return table.concat(self, sep)
    end
    function Collection.new()
        return setmetatable({n = 0}, Collection)
    end
end

local function empty(text)
    -- Return true if text is nil or empty (assuming a string).
    return text == nil or text == ''
end
```



```
local function strip(text)
    -- Return text with no leading/trailing whitespace.
    return text:match("^%s*(.)%s*$")
end

local function normalize(text)
    -- Return text with any strip markers normalized by replacing the
    -- unique number with a fixed value so comparisons work.
    return text:gsub('\\127[^\127]*UNIQ[^\127]*-%)(%x\+)(-QINU[^\127]*\127)'
end

local function status_box(stats, expected, actual, iscomment)
    local label, bgcolor, align, isfail
    if iscomment then
        actual = ''
        align = 'center'
        bgcolor = 'silver'
        label = 'Cmnt'
    elseif expected == '' then
        stats.ignored = stats.ignored + 1
        return '', actual
    elseif normalize(expected) == normalize(actual) then
        stats.pass = stats.pass + 1
        actual = ''
        align = 'center'
        bgcolor = 'green'
        label = 'Pass'
    else
        stats.fail = stats.fail + 1
        align = 'center'
        bgcolor = 'red'
        label = 'Fail'
        isfail = true
    end
    local sbx = 'style="text-align:' .. align .. ';color:white;background:'
    return sbx, actual, isfail
end

local function status_text(stats)
    local bgcolor, ignored_text, msg, ttext
    if stats.template then
        ttext = "''''Using [[Template:" .. stats.template .. "]]:'''"
    else
        ttext = ''
    end
    if stats.fail == 0 then
        if stats.pass == 0 then
            bgcolor = 'salmon'
            msg = 'No tests performed'
        else
            bgcolor = 'green'
            msg = string.format('All %d tests passed', stats.pass)
        end
    else
        bgcolor = 'darkred'
        msg = string.format('%d test%s failed', stats.fail, stats.fail ==
    end
    if stats.ignored == 0 then
        ignored_text = ''
    else
        bgcolor = 'salmon'
        ignored_text = string.format(', %d test%s ignored because expecte
    end
end
```

```
        return ttext .. '<span style="font-size:120%;color:white;background-color:' ..
            msg .. ignored_text .. '</span>'
end

local function run_template(frame, template, args, collapse_multiline)
    -- Template "{{ example | 2 = def | abc | name = ghi jkl }}"
    -- gives xargs { " abc ", "def", name = "ghi jkl" }.
    if template:sub(1, 2) == '{{' and template:sub(-2, -1) == '}}' then
        template = template:sub(3, -3) .. '|' -- append sentinel to get
    else
        return '(invalid template)'
    end
    local xargs = {}
    local index = 1
    local templatename
    local function put_arg(k, v)
        -- Kludge: Module:Val uses Module:Arguments which trims arguments
        -- omits blank arguments. Simulate that here.
        -- LATER Need a parameter to control this.
        if templatename:sub(1, 3) == 'val' then
            v = strip(v)
            if v == '' then
                return
            end
        end
        xargs[k] = v
    end
    end
    template = template:gsub('%[%[[^%[%]]-)|(.-%]|%)', '%1\0%2') -- replace
    for field in template:gmatch('(.-)|') do
        field = field:gsub('%z', '|') -- restore pipe in piped link
        if templatename == nil then
            templatename = args.template or strip(field)
            if templatename == '' then
                return '(invalid template)'
            end
        else
            local k, eq, v = field:match("^(.-)(=)(.*)$")
            if eq then
                k, v = strip(k), strip(v) -- k and/or v can be empty
                local i = tonumber(k)
                if i and i > 0 and string.match(k, '^%d+$') then
                    put_arg(i, v)
                else
                    put_arg(k, v)
                end
            else
                while xargs[index] ~= nil do
                    -- Skip any explicit numbered parameters
                    index = index + 1
                end
                put_arg(index, field)
            end
        end
    end
    end
    if args.test and not xargs.test then
        -- For convert, allow test=preview or test=nopreview to be injected
        -- the convert under test, if it does not already use that parameter
        -- That allows, for example, a preview of make_tests to show nopreview
        xargs.test = args.test
    end
    local function expand(t)
        return frame:expandTemplate(t)
    end
    local ok, result = pcall(expand, { title = templatename, args = xargs })
end
```

```
        if not ok then
            result = 'Error: ' .. result
        end
        if collapse_multiline then
            result = result:gsub('\n', '\\n')
        end
        return result
    end
end

local function _make_tests(frame, all_tests, args)
    local maxlen = 38
    for _, item in ipairs(all_tests) do
        local template = item[1]
        if template then
            local templen = mw.ustring.len(template)
            item.templen = templen
            if maxlen < templen and templen <= 70 then
                maxlen = templen
            end
        end
    end
    local result = Collection.new()
    for _, item in ipairs(all_tests) do
        local template = item[1]
        if template then
            local actual = run_template(frame, template, args, true)
            local pad = string.rep(' ', maxlen - item.templen) .. '
            result:add(template .. pad .. actual)
        else
            local text = item.text
            if text then
                result:add(text)
            end
        end
    end
    end
    -- Pre tags returned by a module are html tags, not like wikitext <pre>.
    return '<pre>\n' .. mw.text.nowiki(result:join('\n')) .. '\n</pre>'
end

local function _run_tests(frame, all_tests, args)
    local function safe_cell(text, multiline)
        -- For testing {{convert}}, want wikitext like '[[kilogram|kg]]'
        -- so the link works and so the displayed text is short (just "kg")
        text = text:gsub('%[%[[^%]]-)|(.-%]])', '%1\0%2') -- replace
        text = text:gsub('{', '&#123;'):gsub('|', '&#124;') -- escape
        text = text:gsub('%z', '|') -- restore
        if multiline then
            text = text:gsub('\\n', '<br />')
        end
        return text
    end
    end
    local function nowiki_cell(text, multiline)
        text = mw.text.nowiki(text)
        if multiline then
            text = text:gsub('\\n', '<br />')
        end
        return text
    end
    end
    local stats = { pass = 0, fail = 0, ignored = 0, template = args.template }
    local result = Collection.new()
    result:add('{| class="wikitable sortable"')
    result:add('! Template !! Expected !! Actual, if different !! Status')
    for _, item in ipairs(all_tests) do
        local template, expected = item[1], item[2] or ''
    end
end
```

```
        if template then
            local actual = run_template(frame, template, args, true)
            local sbox, actual, isfail = status_box(stats, expected,
            result:add('|-')
            result:add('|' .. safe_cell(template))
            result:add('|' .. safe_cell(expected, true))
            result:add('|' .. safe_cell(actual, true))
            result:add('|' .. sbox)
            if isfail then
                result:add('|-')
                result:add('| align="center"| (above, nowiki)')
                result:add('|' .. nowiki_cell(normalize(expected)))
                result:add('|' .. nowiki_cell(normalize(actual)))
                result:add('|')
            end
        end
        else
            local text = item.text
            if text and text:sub(1, 3) == '---' then
                result:add('|-')
                result:add('| colspan="3" style="color:white;background-color:black"|')
                result:add('|' .. status_box(stats, '', '', true))
            end
        end
    end
    result:add('}}')
    return status_text(stats) .. '\n\n' .. result:join('\n')
end

local function get_page_content(page_title, ignore_error)
    local t = mw.title.new(page_title)
    if t then
        local content = t:getContent()
        if content then
            if content:sub(-1) ~= '\n' then
                content = content .. '\n'
            end
            return content
        end
    end
    if not ignore_error then
        error('Could not read wikitext from "[[" .. page_title .. "]"')
    end
end

local function _compare(frame, page_pairs)
    local prefix = frame.args.prefix or '*'
    local function diff_link(title1, title2)
        return '<span class="plainlinks">[[' ..
            tostring(mw.uri.fullUrl('Special:ComparePages',
            { page1 = title1, page2 = title2 })) ..
            ' diff]</span>'
    end
    local function link(title)
        return '[[ ' .. title .. ']]'
    end
    local function message(text, isgood)
        local color = isgood and 'green' or 'darkred'
        return '<span style="color:' .. color .. ';">' .. text .. '</span>'
    end
    local result = Collection.new()
    for _, item in ipairs(page_pairs) do
        local label
        local title1 = item[1]
        local title2 = item[2]
```



```
        if title1 == title2 then
            label = message('same title', false)
        else
            local content1 = get_page_content(title1, true)
            local content2 = get_page_content(title2, true)
            if not content1 or not content2 then
                label = message('does not exist', false)
            elseif content1 == content2 then
                label = message('same content', true)
            else
                label = message('different', false) .. ' (' .. di
            end
        end
        result:add(prefix .. link(title1) .. ' • ' .. link(title2) .. ' '
    end
    return result:join('\n')
end

local function sections(text)
    return {
        first = 1, -- just after the newline at the end of the last head
        this_section = 1,
        next_heading = function(self)
            local first = self.first
            while first <= #text do
                local last, heading
                first, last, heading = text:find('==+[\t ]*([^\n]
                if first then
                    if first == 1 or text:sub(first - 1, first) == '\n' then
                        self.this_section = first
                        self.first = last + 1
                        return heading
                    end
                    first = last + 1
                else
                    break
                end
            end
            self.first = #text + 1
            return nil
        end,
        current_section = function(self)
            local first = self.this_section
            local last = text:find('\n==[^\n]-==[\t\r ]*\n', first)
            if not last then
                last = -1
            end
            return text:sub(first, last)
        end,
    }
end

local function get_tests(frame, tests)
    local args = frame.args
    local page_title, section_title = args.page, args.section
    local show_all = (args.show == 'all')
    if not empty(page_title) then
        if not empty(tests) then
            error('Invoke must not set "page=' .. page_title .. '" if
        end
        if page_title:sub(1, 2) == '[' and page_title:sub(-2) == ']' then
            page_title = strip(page_title:sub(3, -3))
        end
        tests = get_page_content(page_title)
    end
end
```



```
        if not empty(section_title) then
            local s = sections(tests)
            while true do
                local heading = s:next_heading()
                if heading then
                    if heading == section_title then
                        tests = s:current_section()
                        break
                    end
                else
                    error('Section "' .. section_title .. "'
end
                end
            end
        end
    end
end
if type(tests) ~= 'string' then
    if type(tests) == 'table' then
        return tests
    end
    error('No tests were specified; see [[Module:Convert/tester/doc]]
end
if tests:sub(-1) ~= '\n' then
    tests = tests .. '\n'
end
local template_count = 0
local all_tests = Collection.new()
for line in (tests):gmatch('([^\n]-)[\t\r ]*\n') do
    local template, expected = line:match('^({{.-}})%s*(.-)%s*$')
    if template then
        template_count = template_count + 1
        all_tests:add({ template, expected })
    elseif show_all then
        all_tests:add({ text = line })
    end
end
if template_count == 0 then
    error('No templates found; see [[Module:Convert/tester/doc]].', 0
end
return all_tests
end

local function main(frame, p, worker)
    local ok, result = pcall(get_tests, frame, p.tests)
    if ok then
        ok, result = pcall(worker, frame, result, frame.args)
        if ok then
            return result
        end
    end
    return '<strong class="error">Error</strong>\n\n' .. result
end

local modules = {
    -- For convenience, a key defined here can be used to refer to the
    -- corresponding list of modules.
    countries = { -- Commons
        'Countries',
        'Countries/Africa',
        'Countries/Americas',
        'Countries/Arab world',
        'Countries/Asia',
        'Countries/Caribbean',
        'Countries/Central America',
        'Countries/Europe',
```

```
        'Countries/North America',
        'Countries/North America (subcontinent)',
        'Countries/Oceania',
        'Countries/South America',
        'Countries/United Kingdom',
    },
    convert = {
        'Convert',
        'Convert/data',
        'Convert/text',
        'Convert/extra',
        'Convert/wikidata',
        'Convert/wikidata/data',
    },
    cs1 = {
        'Citation/CS1',
        'Citation/CS1/Configuration',
    },
    cs1all = {
        'Citation/CS1',
        'Citation/CS1/Configuration',
        'Citation/CS1/Whitelist',
        'Citation/CS1/Date validation',
    },
    team = {
        'Team appearances list',
        'Team appearances list/data',
        'Team appearances list/show',
    },
    val = {
        'Val',
        'Val/units',
    },
}

local p = {}

function p.compare(frame)
    local page_pairs = p.pairs
    if not page_pairs then
        local args = frame.args
        if not args[2] then
            local builtins = modules[args[1] or 'convert']
            if builtins then
                args = builtins
            end
        end
        end
        page_pairs = {}
        for i, title in ipairs(args) do
            if not title:find(':', 1, true) then
                title = 'Module:' .. title
            end
            page_pairs[i] = { title, title .. '/sandbox' }
        end
    end
    local ok, result = pcall(_compare, frame, page_pairs)
    if ok then
        return result
    end
    return '<strong class="error">Error</strong>\n\n' .. result
end

p.check_sandbox = p.compare
```



```
function p.make_tests(frame)
    return main(frame, p, _make_tests)
end

function p.run_tests(frame)
    return main(frame, p, _run_tests)
end

return p
```