

Modul:Unsubst

Helper module to facilitate a substituted template transform into a template transclusion.

Maintenance templates, such as Vorlage:TI or Vorlage:TI, should never be substituted. A trick to avoid that is to make a template substitute to its transcluded form.

Infoboxes should use Module:Unsubst-infobox, as should any other templates with parameters listed in block format by default.

Usage

To turn a template into a self-substituting template, wrap the existing template code with:

```
{{SAFESUBST:<noinclude />#invoke:Unsubst||$B=  
 [ ... existing template code ... ]  
}}
```

The wikitext to display when not subst'd must be given as "\$B". All other parameters passed to the #invoke will be copied to the generated template invocation as default values. If the value of any of these default parameters is __DATE__, that value in the generated template invocation will be the current month and year.

Some templates have a Vorlage:Tag but no matching Vorlage:Tag at the end of the template. In such cases the missing Vorlage:Tag must be added before the ending Vorlage:)).

Advanced

```
{{SAFESUBST:<noinclude />#invoke:Unsubst||$params=[ parameters ]|$aliases=[ alias  
 [ ... existing template code ... ]  
}}
```

Due to Lua limitations, parameters are normally ordered randomly when the template is substituted. Vorlage:Para can be used in #invoke:Unsubst to list template parameters in order, comma-separated (e.g. egg,bacon,sausage,cheese,spam). Numbered parameters should be before others in the list. Any remaining parameters are tacked onto the end of the generated invocation.

Parameter aliases can be listed in Vorlage:Para (and shouldn't be listed in Vorlage:Para), and will be replaced automatically. Each alias and its replacement should be formatted as alias>replacement, and each of those pairs should be comma-separated (e.g. œuf>egg, melt>cheese). Note that this parameter can function with or without Vorlage:Para.



Parameter **Vorlage:Para** can be used to modify other facets of the module's behaviour; entries are comma-separated. Valid flags are `override` (allows parameters in the `#invoke:` to take precedence over parameters in the original template invocation); `keep-whitespace` (prevents whitespace from being trimmed from unnamed parameters); and `remove-empty` (removes empty parameters).

These parameters can be manipulated using parser functions to provide more complicated options (note that in the parameters any parser function, or template or module invocation, should also have `SAFESUBST:<noinclude />`).

Parameter **Vorlage:Para** will override the subst'd templates name with the template name assigned to this parameter.

Example

Consider a template `Template:Example` containing the following code:

```
{{SAFESUBST:<noinclude />#invoke:Unsubst|foo=bar |date=__DATE__ |$B=
 [ ... Template code goes here ... ]
}}
```

Original	Result
Vorlage:Tlsc	Vorlage:Tlc
Vorlage:Tlsc	Vorlage:Tlc
Vorlage:Tlsc	Vorlage:Tlc
Vorlage:Tlsc	Vorlage:Tlc

```
local checkType = require('libraryUtil').checkType
local p = {}
local BODY_PARAM = '$B'
local specialParams = {
    ['$params'] = 'parameter list',
    ['$aliases'] = 'parameter aliases',
    ['$flags'] = 'flags',
    ['$B'] = 'template content',
    ['$template-name'] = 'template invocation name override',
}
function p.main(frame, body)
    -- If we are substing, this function returns a template invocation, and if
    -- not, it returns the template body. The template body can be specified
    -- the body parameter, or in the template parameter defined in the
    -- BODY_PARAM variable. This function can be called from Lua or from
    -- #invoke.
```

```
-- Return the template body if we aren't substing.
if not mw.isSubsting() then
    if body ~= nil then
        return body
    elseif frame.args[BODY_PARAM] ~= nil then
        return frame.args[BODY_PARAM]
    else
        error(string.format(
            "no template content specified (use parameter '%s'",
            BODY_PARAM
        ), 2)
    end
end

-- Sanity check for the frame object.
if type(frame) ~= 'table'
    or type(frame.getParent) ~= 'function'
    or not frame:getParent()
then
    error(
        "argument #1 to 'main' must be a frame object with a parent
        "frame available",
        2
    )
end

-- Find the invocation name.
local mTemplateInvocation = require('Module:Template invocation')
local name

if frame.args['$template-name'] and '' ~= frame.args['$template-name'] then
    name = frame.args['$template-name']
else
    name = mTemplateInvocation.name(frame:getParent():getTitle())
end

-- Combine passed args with passed defaults
local args = {}
if string.find( '..',(frame.args['$flags'] or '')..' ',',', '%s*override%s*'
    for k, v in pairs( frame:getParent().args ) do
        args[k] = v
    end
    for k, v in pairs( frame.args ) do
        if not specialParams[k] then
            if v == '__DATE__' then
                v = mw.getContentLanguage():formatDate(
            end
            args[k] = v
        end
    end
end
else
    for k, v in pairs( frame.args ) do
        if not specialParams[k] then
            if v == '__DATE__' then
                v = mw.getContentLanguage():formatDate(
            end
            args[k] = v
        end
    end
    for k, v in pairs( frame:getParent().args ) do
        args[k] = v
    end
end
end
```

```
-- Trim parameters, if not specified otherwise
if not string.find( '..',(frame.args['$flags'] or '')..' ', '%s*keep%-w'
    for k, v in pairs( args ) do args[k] = mw.ustring.match(v, '^%s*(
end

-- Pull information from parameter aliases
local aliases = {}
if frame.args['$aliases'] then
    local list = mw.text.split( frame.args['$aliases'], '%s*,%s*' )
    for k, v in ipairs( list ) do
        local tmp = mw.text.split( v, '%s*>%s*' )
        aliases[tonumber(mw.ustring.match(tmp[1], '^([1-9][0-9]*$'
    end
end
for k, v in pairs( aliases ) do
    if args[k] and ( not args[v] or args[v] == '' ) then
        args[v] = args[k]
    end
    args[k] = nil
end

-- Remove empty parameters, if specified
if string.find( '..',(frame.args['$flags'] or '')..' ', '%s*remove%-empt
    local tmp = 0
    for k, v in ipairs( args ) do
        if v ~= '' or ( args[k+1] and args[k+1] ~= '' ) or ( args
            tmp = k
        else
            break
        end
    end
    for k, v in pairs( args ) do
        if v == '' then
            if not (type(k) == 'number' and k < tmp) then arg
        end
    end
end

-- Order parameters
if frame.args['$params'] then
    local params, tmp = mw.text.split( frame.args['$params'], '%s*,%s
    for k, v in ipairs(params) do
        v = tonumber(mw.ustring.match(v, '^([1-9][0-9]*$')) or v
        if args[v] then tmp[v], args[v] = args[v], nil end
    end
    for k, v in pairs(args) do tmp[k], args[k] = args[k], nil end
    args = tmp
end

return mTemplateInvocation.invocation(name, args)
end

p['] = p.main -- For backwards compatibility
return p
```