



Inhaltsverzeichnis

Modul:Item

This template implements [Vorlage:TI](#), [Vorlage:TI](#) and [Vorlage:TI](#).

```
local p = {}

local function escape(str)
    return str:gsub("[|\\]", function (c) return string.format("\\%03d", c:byte()) end)
end

local function unescape(str)
    return str:gsub("\\(%d%d%d)", function (d) return string.char(d) end)
end

-- Implements [[Template:Item]]
function p.pack(frame)
    local parent = frame:getParent()
    local result = ''
    for key, value in pairs(parent.args) do
        result = result .. "|" .. escape(tostring(key)) .. "|" .. escape(tostring(value))
    end
    return result .. "|";
end

local function unpack(str)
    local result = {}
    for key, value in str:gfind("|([^\|]*)|([^\|]*)") do
        result[unescape(key)] = unescape(value)
    end
    return result
end

-- Implements [[Template:Component]]
function p.component(frame)
    return unpack(frame.args[1])[frame.args[2]]
end

local function getItems(frame)
    return frame:getParent().args
end

local function invert(tbl)
    local result = {}
    for key, value in pairs(tbl) do
        result[value] = key
    end
    return result
end

-- Add args into item as appropriate (see [[Template:Format item]])
local function addArgs(
    item, -- unpacked item to modify
    args, -- arguments for adding into item
    ignore, -- pass in invert{keys to ignore}
    shift -- for numbered arguments, args[key+shift] is assigned to item[key]
)
    -- returns: item
    for key, value in pairs(args) do
        if not ignore[key] then
            item[key] = value
        end
    end
end
```



```
        local _, _, paramKey = string.find(key, "^param (.*)")
        local _, _, importantKey = string.find(key, "^important ")
        paramKey = paramKey or importantKey or key
        if shift and type(paramKey) == "number" then
            paramKey = paramKey - shift
            if paramKey < 1 then paramKey = nil end
        end
        if paramKey and (importantKey or item[paramKey] == nil) then
            item[paramKey] = value
        end
    end
end

return item
end

-- Implements [[Template:Format item]]
function p.format(frame)
    local args = frame:getParent().args
    local ignore = invert{ "template", "item" }
    local templateArgs = addArgs(unpack(args.item), args, ignore)

    return frame:expandTemplate{ title = args.template, args = templateArgs }
end

-- See [[Template:Item#Format each item using a template]]
function p.each(frame)
    local args = frame.args
    local items = getItems(frame)
    local separator = args[1] or ""
    local prepend = args[2] or ""
    local append = args[3] or ""
    local ignore = invert{ "template" }
    local shift = 3

    local result = ""
    for i, item in ipairs(items) do
        local templateArgs = addArgs(unpack(item), args, ignore, shift)
        result = result .. prepend .. frame:expandTemplate{ title = args.template, args = templateArgs }
        if items[i + 1] then
            result = result .. separator
        end
    end
    return result
end

-- See [[Template:Item#Gather given parameter from all items]]
function p.gather(frame)
    local args = frame.args
    local items = getItems(frame)
    local parameter = args.parameter or "1"

    local templateArgs = { }
    for i, item in ipairs(items) do
        templateArgs[i] = unpack(item)[parameter]
    end

    return frame:expandTemplate{ title = args.template, args = templateArgs }
end

return p
```