Modul:TNTTools

Contains functions linked to Module:TNT, which at the same time make calls to multilingual tables, located in Commons, for the creation of modules and multilingual templates.

TNTTools has:

- **Question functions**: with boolean or numerical indexed return. To be called from other modules or from templates. With:
  - Case sensitive option.
  - Possibility of **more than one translated text value** (where each value is separated by "|").
- To put aside write, adding "I18n/" as a prefix and ".tab" extension as a suffix for the table names.

```
local p = {}

local TNT = require('Module:TNT')
--local SD = require('Module:SimpleDebug')

function p.TNTTabFull (TNTTab)
        if (string.sub(TNTTab, 1, 5)) ~= 'I18n/' then
                TNTTab = 'I18n/'..TNTTab
        end
        if (string.sub(TNTTab, string.len(TNTTab)-3)) ~= '.tab' then
                TNTTab = TNTTab..'.tab'
        end
        return TNTTab
end --TNTTabFull

function p.TNTTabCommons (TNTTab)
        return 'Commons:Data:'..p.TNTTabFull(TNTTab)
end

function p.LnkTNTTab (TNTTab)
        return '[['..p.TNTTabCommons(TNTTab)..']]'
end

function I18nStr (TNTTab, S, IsMsg, params)
        TNTTab = p.TNTTabFull (TNTTab)
        local SEnd = TNT.format(TNTTab, S, unpack(params)) or ''
        if SEnd == '' then
                SEnd = TNT.formatInLanguage('en',TNTTab, S, unpack(params))
                if IsMsg then
                        local icon = '[[File:Arbcom ru editing.svg|12px|Not found
                        SEnd = SEnd..icon
                end
        end
        return SEnd
end --I18nStr

function p.GetMsgP (TNTTab, S, ...)
        return I18nStr (TNTTab, S, true, {...})
end
```

```
function p.GetStrP (TNTTab, S, ...)
        return I18nStr (TNTTab, S, false, {...})
end

function p.TabTransCS (TNTTab, S, CaseSensitive)
        CaseSensitive = ((CaseSensitive ~= nil) and (CaseSensitive == true)) or t
        local Wds = TNT.format (p.TNTTabFull(TNTTab), S)
        if not CaseSensitive then
                Wds = string.lower (Wds)
        end
    return mw.text.split (Wds, '|')
end --TabTransCS

function p.TabTransMT (TNTTab, S, MaxTrans)
        local FN = p.TNTTabFull(TNTTab)
        local tab = mw.text.split (TNT.format (FN, S), '|')
        if #tab > MaxTrans then
                error (string.format('Found %s translations for "%s". Search in |
                                                        -- Translation not requi
        end
        return tab
end --TabTransMT

function p.SFoundInTNTArr (TNTTab, val, CaseSensitive, S)
        if (S == nil) or (S == '') then
                error('Not arguments trying to find "'..val..'"') --It doesn't re
        end
        local Arr = p.TabTransCS (TNTTab, S, CaseSensitive)
        if not CaseSensitive then
                val = string.lower (val)
        end
        for I, W in ipairs(Arr) do
                if W == val then
                        return true
                end
        end
        return false
end --SFoundInTNTArr

function p.IdxFromTabTrans (TNTTab, val, CaseSensitive, ...)
        local Arr = unpack(arg)
        if Arr == nil then
                error('Not arguments trying to find "'..val..'"') --It doesn't re
        end
        local Idx = 0
        for I, W in ipairs(Arr) do
                if p.SFoundInTNTArr (TNTTab, val, CaseSensitive, W) then
                        Idx = I
                        break
                end
        end
        return Idx
end        --IdxFromTabTrans

return p
```