Navigation

Tools

Personal tools

personal-extra

Random page

Views

Actions

Module:Delink

From Blue Gold Program Wiki
The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.

Documentation for this module may be created at Module:Delink/doc

```
-- This module de-links most wikitext.

require('Module:No globals')

local p = {}

local getArgs

local function delinkReversePipeTrick(s)
    if s:match("^%[%[|.*[|\n]") then -- Check for newlines or multiple pipes.
        return s
    else
        return s:match("%[%[|(.*)%]%]")
    end
end

local function delinkPipeTrick(s)
    local linkarea, display = "", ""
    -- We need to deal with colons, brackets, and commas, per [[Help:Pipe
trick]].
    -- First, remove the text before the first colon, if any.
    if s:match(":") then
        s = s:match("%[%[.-:(.*)|%]%]")
    -- If there are no colons, grab all of the text apart from the square
brackets and the pipe.
    else
        s = s:match("%[%[(.*)|%]%]")
    end
    -- Next up, brackets and commas.
    if s:match("%(.-%)$") then -- Brackets trump commas.
        s = s:match("(.-) ?%(.-%)$")
    elseif s:match(",") then -- If there are no brackets, display only the
text before the first comma.
        s = s:match("(.-),.*$")
    end
    return s
end

local function delinkWikilink(s)
    local result = s
    -- Deal with the reverse pipe trick.
    if result:match("%[%[|") then
        return delinkReversePipeTrick(result)
    end
    result = mw.uri.decode(result, "PATH") -- decode percent-encoded
entities. Leave underscores and plus signs.
    result = mw.text.decode(result, true) -- decode HTML entities.
    -- Check for bad titles. To do this we need to find the
    -- title area of the link, i.e. the part before any pipes.
    local titlearea
    if result:match("|") then -- Find if we're dealing with a piped link.
```

```
        titlearea = result:match("^%[%[(.-)|.*%]%]")
    else
        titlearea = result:match("^%[%[(.-)%]%]")
    end
    -- Check for bad characters.
    if mw.ustring.match(titlearea, "[%[%]<>{}%%□%c\n]") then
        return s
    end
    -- Check for categories, interwikis, and files.
    local colonprefix = result:match("%[%[(.-):.*%]%]") or "" -- Get the text
before the first colon.
    local ns = mw.site.namespaces[colonprefix] -- see if this is a known
namespace
    if mw.language.isKnownLanguageTag(colonprefix)
    or ( ns and ( ns.canonicalName == "File" or ns.canonicalName ==
"Category" ) ) then
        return ""
    end
    -- Remove the colon if the link is using the [[Help:Colon trick]].
    if result:match("%[%[:") then
        result = "[[" .. result:match("%[%[:(.*%]%])")
    end
    -- Deal with links using the [[Help:Pipe trick]].
    if mw.ustring.match(result, "^%[%[[^|]*|%]%]") then
        return delinkPipeTrick(result)
    end
    -- Find the display area of the wikilink
    if result:match("|") then -- Find if we're dealing with a piped link.
        result = result:match("^%[%[.-|(.+)%]%]")
        -- Remove new lines from the display of multiline piped links,
        -- where the pipe is before the first new line.
        result = result:gsub("\n", "")
    else
        result = result:match("^%[%[(.-)%]%]")
    end

    return result
end

local function delinkURL(s)
    -- Assume we have already delinked internal wikilinks, and that
    -- we have been passed some text between two square brackets [foo].
    -- If the text contains a line break it is not formatted as a URL,
regardless of other content.
    if s:match("\n") then
        return s
    end
    -- Check if the text has a valid URL prefix and at least one valid URL
character.
    local valid_url_prefixes = {"//", "http://", "https://", "ftp://",
"gopher://", "mailto:", "news:", "irc://"}
```

```
    local url_prefix
    for i,v in ipairs(valid_url_prefixes) do
        if mw.ustring.match(s, '^%[' .. v ..'[^"%s□].*%]' ) then
            url_prefix = v
            break
        end
    end
    -- Get display text
    if not url_prefix then
        return s
    end
    s = s:match("^%[" .. url_prefix .. "(.*)%]") -- Grab all of the text
after the URL prefix and before the final square bracket.
    s = s:match('^.-(["<> □].*)') or "" -- Grab all of the text after the
first URL separator character ("<> ).
    s = mw.ustring.match(s, "^%s*(%S.*)$") or "" -- If the separating
character was a space, trim it off.
    local s_decoded = mw.text.decode(s, true)
    if mw.ustring.match(s_decoded, "%c") then
        return s
    else
        return s_decoded
    end
end

local function delinkLinkClass(s, pattern, delinkFunction)
    if not type(s) == "string" then
        error("Attempt to de-link non-string input.", 2)
    end
    if not ( type(pattern) == "string" and mw.ustring.sub(pattern, 1, 1) ==
"^" ) then
        error('Invalid pattern detected. Patterns must begin with "^".', 2)
    end
    -- Iterate over the text string, and replace any matched text. using the
    -- delink function. We need to iterate character by character rather
    -- than just use gsub, otherwise nested links aren't detected properly.
    local result = ""
    while s ~= '' do
        -- Replace text using one iteration of gsub.
        s = mw.ustring.gsub(s, pattern, delinkFunction, 1)
        -- Append the left-most character to the result string.
        result = result .. mw.ustring.sub(s, 1, 1)
        s = mw.ustring.sub(s, 2, -1)
    end
    return result
end

function p._delink(args)
    local text = args[1] or ""
    if args.refs == "yes" then
        -- Remove any [[Help:Strip markers]] representing ref tags. In most
```

```
situations
        -- this is not a good idea - only use it if you know what you are
doing!
        text = mw.ustring.gsub(text, "UNIQ%w*%-ref%-%d*%-QINU", "")
    end
    if not (args.comments == "no") then
        text = text:gsub("<!%-%-.-%-%->", "") -- Remove html comments.
    end
    if not (args.wikilinks == "no") then
        text = delinkLinkClass(text, "^%[%[.-%]%]", delinkWikilink) -- De-
link wikilinks.
    end
    if not (args.urls == "no") then
        text = delinkLinkClass(text, "^%[.-%]", delinkURL) -- De-link URLs.
    end
    if not (args.whitespace == "no") then
        -- Replace single new lines with a single space, but leave double new
lines
        -- and new lines only containing spaces or tabs before a second new
line.
        text = mw.ustring.gsub(text, "([^\n \t][ \t]*)\n([ \t]*[^\n \t])",
"%1 %2")
        text = text:gsub("[ \t]+", " ") -- Remove extra tabs and spaces.
    end
    return text
end

function p.delink(frame)
    if not getArgs then
        getArgs = require('Module:Arguments').getArgs
    end
    return p._delink(getArgs(frame, {wrappers = 'Template:Delink'}))
end

return p
```


Namespaces

- Module
- Discussion

Variants

This page was last edited on 19 February 2020, at 09:51.

Blue Gold Program Wiki

The wiki version of the Lessons Learnt Report of the Blue Gold program, documents the experiences of a technical assistance (TA) team working in a development project implemented by the

Bangladesh Water Development Board (BWDB) and the Department of Agricultural Extension (DAE) over an eight+ year period from March 2013 to December 2021. The wiki lessons learnt report (LLR) is intended to complement the BWDB and DAE project completion reports (PCRs), with the aim of recording lessons learnt for use in the design and implementation of future interventions in the coastal zone.