Module:Catalog lookup link

Documentation for this module may be created at Module:Catalog lookup link/doc

```
--[[
|1=, |2=, |3=, |4=, |5=, |6=, |7=, |8=, |9=: Optional unnamed parameters for 0 to 9 items to be listed.
        Whitespace is trimmed off both ends and the strings are urlencoded as if they were query strings.
|article-link=: Optional Wikipedia article name to link to.
|article-name=: Optional alternative text to be displayed for |article-link= link in front of catalog link.
        If not specified, |article-link= is used for display as well. If both parameters are not specified, the prefix is omitted completely.
|article-suffix=: Optional symbol to be displayed after article name or link (f.e. ":"; omitted, if not defined).
|link-prefix=: Optional prefix portion of url to external catalog item(s).
|link-suffix=: Optional suffix portion of url to external catalog item(s).
|item-prefix=: Optional text displayed in front of each external link (omitted, if not defined)
|item-suffix=: Optional text displayed immediately after each external link (omitted, if not defined)
|list-separator=: Optional alternative separator displayed between list items (default: ", ", if not specified). Whitespace must be encoded.
|list-leadout=: Optional alternative leadout text displayed between the last two list items (f.e. "and", "or", "as well as", etc., default is the |list-separator= or ", ".)
|leadout-suffix=: Optional alternative suffix text of the leadout (see |list-leadout=) displayed between the last two list items.
        This gets added in front of the last list item instead of the default whitespace which is added without this parameter.
        This may be necessary if |list-separator= is used not only to define the list separator but also parts of the item prefix
        (except for the first one). (At present, this is used only to cope with format oddities of the {{MR}} template.)


new parameters that support access icons:
|allowed_icons= – comma-separated list of keywords: free, limited, registration, subscription, none, all (default; 'all' implied when this parameter empty or omitted)
        the icons specified in the following parameters are checked agains the list in |allowed-icons=; not in the list? not displayed
|url-access-all= – applies specified icon to all items in the list; accepted keywords: free, limited, registration, subscription;
|url-accessn= – applies specified icon to item n of the list (the nth positional parameter); accepted keywords: free, limited, registration, subscription;

]]

require('Module:No globals');
local getArgs = require ('Module:Arguments').getArgs;
local lock_icons = {
--icon classes are defined in Module:Citation/CS1/styles.css
```

```
        ['free'] = {'cs1-lock-free', 'Freely accessible'},
        ['registration'] = {'cs1-lock-registration', 'Free registration
required'},
        ['limited'] = {'cs1-lock-limited', 'Free access subject to limited
trial, subscription normally required'},
        ['subscription'] = {'cs1-lock-subscription', 'Paid subscription
required'},
        }


--[[--------------------------< I S _ S E T >------------------------------
----------------------------------

Returns true if argument is set; false otherwise. Argument is 'set' when it
exists (not nil) or when it is not an empty string.

]]

local function is_set( var )
        return not (var == nil or var == '');
end


--[=[-------------------------< M A K E _ L A B E L >----------------------
----------------------------------

Makes a wikilinked or plain text label from arguments; when both link and
display text is provided, makes a
wikilink in the form [[L|D]]; if only link is provided, makes a wikilinked
label in the form [[L]]; if only display
is provided, makes a plain-text label; if neither are provided makes a label
from suffix, returns an empty string else.

]=]

local function make_label (link, display, suffix)
local label = '';
        if is_set (link) then
                if is_set (display) then
                        label = table.concat ({'[[', link, '|', display,
']]'});                         -- make [[L|D]] wikilinked label
                else
                        label = table.concat ({'[[', link, ']]'});
-- make [[L]] wikilinked label
                end
        elseif is_set (display) then
                label = display;
-- plain-text label
        end

        if is_set (label) then
```

```
                return table.concat ({label, suffix, ' '});
-- assemble the complete label
        else
                return suffix;
-- no space after suffix if no label
        end
end


--[[--------------------------< I C O N _ I N D E X _ G E T >---------------
---------------------------------

returns index into lock_icons[] if value assigned to |url-access= or |url-
access-all= is a valid icon selector
(free, limited, registration, subscription)

icon selection may be limited to a subset of the icons with:
        |allow_icons=<comma-separated list of allowed icons>
<comma-separated list of allowed icons> may be any of the keywords: free,
limited, registration, subscription, none, all

keyword 'all' is default condition; 'all' is implied when |allowed=icons= is
empty or omitted

keyword 'none' for use with identifiers where icons are inappropriate (isbn,
issn, oclc)

Templates using this module should set:
        |allow_icons=free for most identifiers;
        |allow_icons=none for isbn, issn, oclc, etc

|url-access= is alias of |url-access1=

]]

local function icon_index_get (args, k)
        local icon;
        local param_name = (1 == k and is_set (args['url-access']) and 'url-
access') or table.concat ({'url-access', k});        -- make an enumerated
parameter name

        if is_set (args['url-access-all']) and lock_icons[args['url-access-
all']] then        -- if set and valid
                icon = args['url-access-all'];
-- tentatively

        elseif is_set (args[param_name]) and lock_icons[args[param_name]]
then               -- if set and valid
                icon = args[param_name];
-- tentatively
```

```
        else
                return nil;
-- neither |url-access-all= nor |url-accessn= set so return nil
        end

        if args['allow_icons'] and args['allow_icons']:find ('none') then
-- if 'none' keyword is present
                return nil;
-- icons display not allowed
        end

        if not is_set (args['allow_icons']) or args['allow_icons']:find
('all') or args['allow_icons']:find (icon) then        --if all allowed or
specified icon is allowed
                return icon;
-- return selected icon as index into icon table
        end
end


--[[--------------------------< M A I N >----------------------------------
----------------------------------

Template entrypoint to this module; arguments come primarily from the parent
frame though in templates that use
this module, |allowed-icons= is typically set, if needed, in the
{{#invoke:}}.

]]

local function main (frame)
        local args = getArgs (frame);
        local out_text = '';

        if is_set(args[1]) then
                local result = {};
                local label;
                local article_suffix = args['article-suffix'] or
args['article-postfix'] or '';
                local link_prefix = args['link-prefix'] or '';
                local link_suffix = args['link-suffix'] or args['link-
postfix'] or '';
                local item_prefix = args['item-prefix'] or '';
                local item_suffix = args['item-suffix'] or args['item-
postfix'] or '';
                local list_separator = args['list-separator'] or ', ';
                local leadout_suffix = args['leadout-suffix'] or
args['leadout-postfix'] or ' ';
                local list_leadout;

                local icon_index;
```

```
                if is_set (args['list-leadout']) then
                        list_leadout = table.concat ({
                                mw.ustring.gsub (args['list-leadout'],
'^(%a)', ' %1'),                        -- insert leading space if first
character is a letter
                                leadout_suffix,
                        });
                else
                        list_leadout = '';
                end
                label = make_label (args['article-link'], args['article-
name'], article_suffix);

                for k, item in ipairs (args) do
-- for each of the positional parameters
                        item = mw.text.trim (item);
-- remove extraneous whitespace
                        if is_set (link_prefix) then
-- if there is link prefix...
                                item = table.concat ({
-- create an external link item
                                        '[',
-- open ext link markup
                                        link_prefix,
-- url prefix
                                        mw.uri.encode (item),
-- item is part of url
                                        link_suffix,
-- url suffix
                                        ' ',
-- required space between url and label
                                        item_prefix,
-- label prefix
                                        item,
-- item as label
                                        item_suffix,
-- item suffix
                                        ']'
-- close ext link markup
                                });

                                icon_index = icon_index_get (args, k);
-- set if icon specified and allowed for this item; nil else
                                if icon_index then
                                        item = table.concat ({
-- add access icon markup to this item
                                                '<span class="',
-- open the opening span tag; icon classes are defined in
Module:Citation/CS1/styles.css
                                                lock_icons[icon_index][1],
-- add the appropriate lock icon class
```

```
                                                '" title="',
-- and the title attribute
                                                lock_icons[icon_index][2],
-- for an appropriate tool tip
                                                '">',
-- close the opening span tag
                                                item,
                                                '</span>',
-- and close the span
                                        });
                                end
                        else
                                item = table.concat ({
-- create an unlinked item
                                        item_prefix,
-- label prefix
                                        item,
-- item as label
                                        item_suffix,
-- item suffix
                                });
                        end
                        table.insert (result, item);
-- add the item to the result list
                end
                if is_set (args['list-leadout']) then
                        out_text = table.concat ({label, mw.text.listToText
(result, list_separator, list_leadout)});
                else
                        out_text = table.concat ({label, table.concat
(result, list_separator)});
                end

        end        --is_set (args[1])
        return out_text
end

return {main = main};
```


Namespaces

- Module
- Discussion

Variants

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.