External Search Part

Table of Contents

Contents

Overview

To have a generic way of searching external sources (repositories, archives, search engines, databases, etc.) the search_external part is created that handles the overall search configuration and presentation and handling, while delegating the actual search implementation specific to the lower level library functions.

Plug-In concept

Introduction

The search_external part needs to know which kinds of external search implementations are available for a system. This is handled by adding a specific configuration entry to the session settings, further described in the configuration paragraph below. Additionally, each plug-in will have to conform to a generic interface that specifies which function(s) to call and which data is passed to and from the plug-in. This is laid out in the Data structure paragraph.

Configuration

The search_external part looks for the following configuration structure:

```
_session = {
   search_plugins = {
      some_plugin = {
         caption = "Title of the plugin",
         default_uri = "https://www.somewhere.com/",
         icon = "https://www.somewhere.com/favicon.ico",
         plugin = "ws/someplugin",
      },
   }
}
```

Each plugin is a datastructure indexed by a unique name in the _session.search_plugins data structure. The caption is the text shown in the dropdown selection of the search_external part. The default_uri is the default external address to be used (which can be overridden by the generic configuration). The plugin setting is the path to the actual plugin api. The icon specifies the path to a visualization of the specific search implementation; currently not used as the icon cannot be displayed easily in the standard HTML select control.

Input parameters

The url is the configured url to use. The searchterm is the term to search for. The parameters are the plug-in specific parameters. And finally, the ctx is the context where data is to be retrieved from by the plug-in (for interpolation for example).

Returned result

When the search call fails, the API is expected to return:

```
nil, 'Failure reason description'
```

When search call succeeds, the API is expected to return 4 values:

```
search_results, more_results_available, total_results, total_pages
```

- The first is the actual object containing all search results.

- The more_results_available is a flag that indicates if there are more results than currently returned in the search_results object.
- Regardless if paging is used, the total_results should contain the total number of available results (which can be equal or more than the results in total_results).
- And lastly, the total_pages is the number of available result pages there might be. If all results are in search_results, total_pages is expected to be 1 and more_results_available to be nil and total_results to be equal to the number of results in search_results.

The search_results object has the following properties:

Property	Type	Description
id	String / number	This is a 'generic' id used by the write/read cycle of the search_external part to detemine which item was selected; id assumed to be unique;
link	String	This is a URL that can be used to open a new web page further detailing the search result; no processing of the URL is done by search_external part;
title	String	The title to be displayed for the search result;
content	String	The actual search result content;
creation_date	Any	Example optional property

The id, link, title and content are mandatory properties, as they are used for displaying the search results. The link, title and content are also HTML stripped to prevent corrupted layouts or XSS.

Plug-in specific parameters

Normally, the url and searchterm parameters will not be enough information for a search API to be fully functional. Therefor the parameters object can be used to pass the plug-in specific details. The following chapter will describe how this object can be created automatically by the configuration of the search_external part.

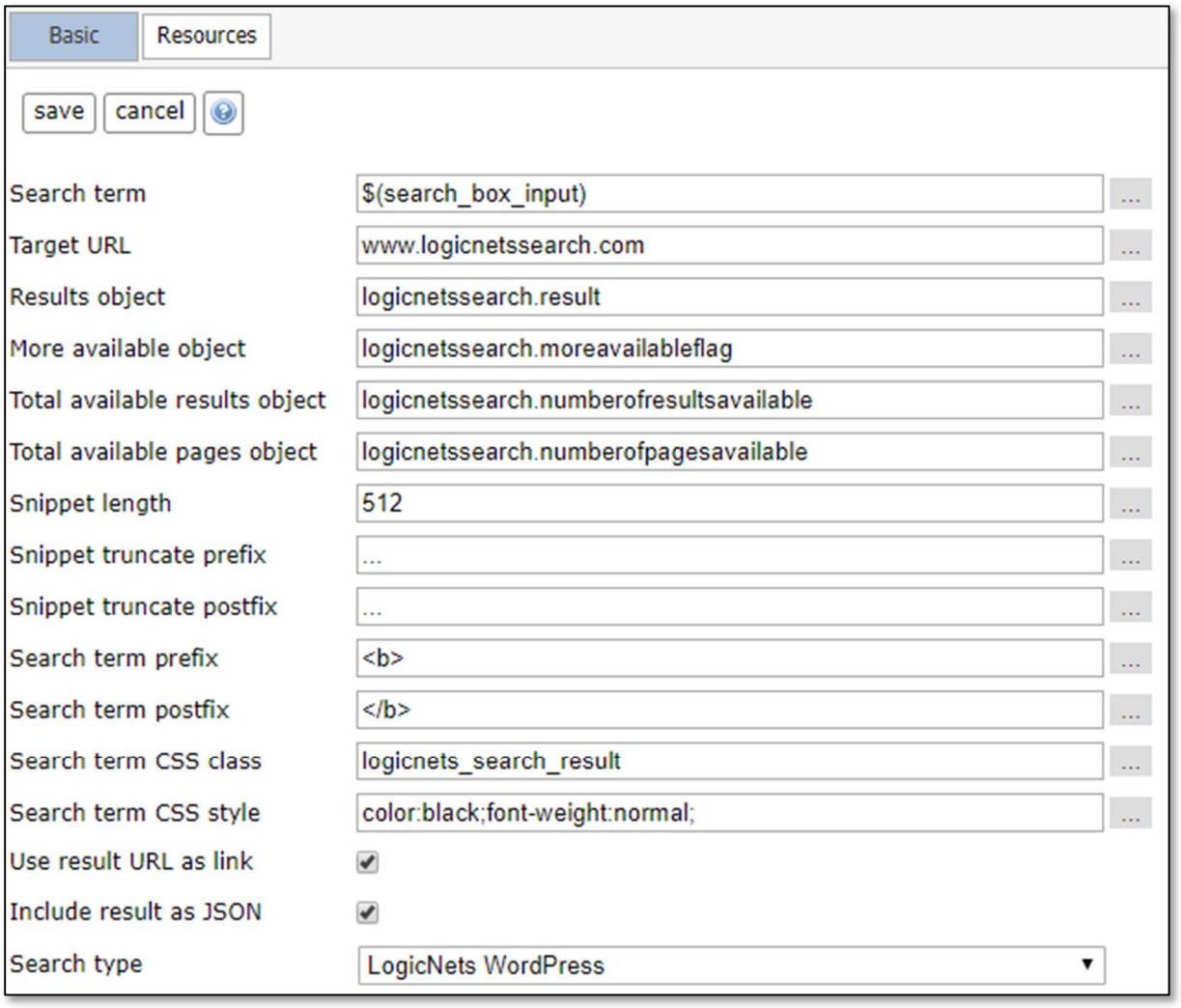
Part configuration

The part configuration consists of the following components:

- First the 'generic' configuration, all related to what to search and how to process the search results.
- Secondly, the specific 'plug-in' configuration where the required items for the plug-in can be set. The following paragraphs describe this in further detail.

Generic configuration

As an example, the generic configuration looks like (when using in a form(part)):



The [Search term] and [Target URL] match the input parameters for the specific API: searchterm and url respectively. The following four match the returned data from the search call.

These are followed by seven styling configuration options which determine the look and feel of the results.

The [Snippet length] determines the maximum length of the content of the result in number of characters. If not filled in, then default of 256 is used. To prevent truncation, use a negative number (or insanely large number).

The truncation of the content is done in a manner to have the first search term at least in the truncated result. Thus, when the beginning of the content is stripped away, the [Snippet truncate prefix] is added as prefix, indicating truncation. The same for the postfix, e.g. when the end is stripped, then the postfix is added. The default is '…' for both prefix and postfix. The value can be anything, including HTML.

To be able to style the search terms themselves, there are several different possibilities. First the prefix and postfix, these (like the snippet truncate) are added before and after the search term. When not specified, default prefix is '' and '' for the search term postfix. But it is also possible to add a CSS style class and/or style definition. As a result, the search term is styled like:

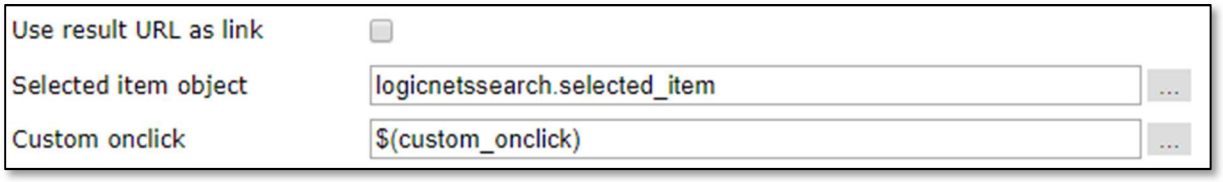
```
<span class="$(css_class)" style="$(css_style)">$(term_prefix)$(term)$(term_postfix)</span>
```

When no CSS style class and no CSS style definition are configured, then the entire wrapping is left out, like:

```
$(term_prefix)$(term)$(term_postfix)
```

The checkbox [Use result URL as link] determines how the title of the result is shown. When this is checked, then a <a> link is created which opens the url directly in a different tab. In that case, the url of the result must be a fully valid resolvable URL.

When not checked, then two additional configuration options become available:



The [Selected item object] can be used to store the selected result into. With the [Custom onclick] it is possible to define custom event handling. When the custom event handler is not configured, then default auto-submit handling is added to the on click event. In case the custom event handler is configured, then the auto-submit handling is omitted, e.g. the custom event handler needs to do this by itself if needed.

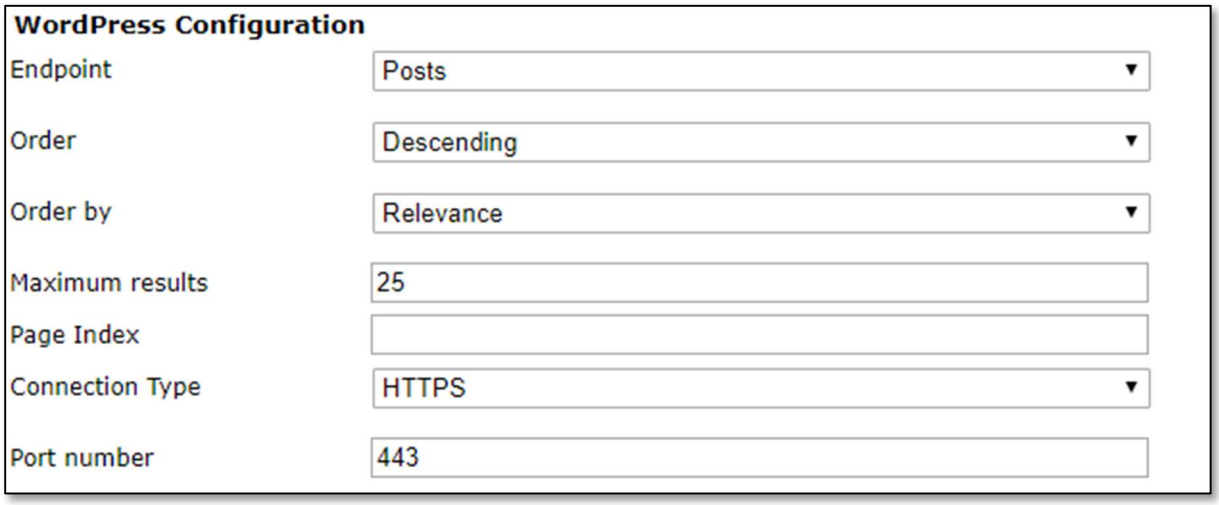
To allow plug-in specific data to be passed to the client side, the checkbox [Include result as JSON] can be checked, converting each entire result into a JSON structure which is printed (hidden) to the client side. If the result structure contains a lot of data, then a lot of data will be sent to the client!

NOTE: When the part is used in a process node, then the visualization options are hidden, like:

The bottom configuration item [Search type] defines which search plug-in to use. The list is filled with the captions of the search_plugins configuration. There is no default, the first available will be auto-selected. When there are no plug-ins configured (or no licenses), then 'None available' is shown and the search will be unusable.

Plug-in configuration

When a [Search type] is selected, the API specific configuration is shown. For example, the WordPress integration has these configuration elements:



The API implementation needs to define the meta structure like a part meta structure. The definition will be taken over for each specified tab. If the caption of the tabs does not match the caption of the tabs of the search_external part, then the configuration items are NOT taken over. There currently is no nice way of creating dynamic tabs in the part editor.

Each configuration item that needs to be passed into the search parameters structure, need to be defined as part of the substructure with the name plugin_params. The entire plugin_params structure will be passed as the parameters structure.

Visualization

Basic

The search result visualization is modelled to be comparable to the freetextsearch part. The results are printed into the following HTML structure:

```
<div class="search_results_container $(container_results_style_class)/$(container_no_results_style_class)">
   <ul class="search_result $(individual_result_style_class)">
      <li class="search_result_title">
         <a href="javascript:void(0);" class="search_result_link" onclick="...">
            Example <span class="example" style="color:OrangeRed;font-weight:bold;">[Search]</span> result
         </a>
      </li>
      <li class="search_result_snippet">
         Example <span class="example" style="color:OrangeRed;font-weight:bold;">[Search]</span> result
      </li>
      <div class="search_result_as_json" style="display: none;">
         {"id":1155,"content":"Example Search result","modified_date_utc":"2019-04-15T14:45:31","title":"Example Search
result","link":"https://www.logicnets.com/","author":97}
      </div>
   </ul>
   <input type="hidden" name="40_1_1-40-search-external--1_selected_result" id="40_1_1-40-search-external--1_selected_result"
value="">
</div>
```

All results are wrapped in a with class [search_results_container]. Each result is wrapped in an <ul> with class [search_result]. This in turn is split up into two 's for the title and for the snippet, with classes [search_result_title] and [search_result_snippet]. A hidden is added with the result as a JSON structure. After the last <ul> search result, the hidden input is added to store the selected item into.

The basic visualization can be expanded with custom CSS classes. For example, either the **Container results CSS class** or the **Container no results CSS** class is added to the search result container depending if there are search results.

Custom templates

The search external part visualization can be customized in the Style tab by using templates for the wrapping container (the information generic to the search) (setting: **Container results template location**) and for the individual search results (setting: **Result template location**). Additionally, a specific result template (setting: **No result template location**) can be used for styling a message when no search result is found.

The custom templates can use interpolation to access the information from the search results. For the container template, the following fields are usable when search results are available:

Field for container template	Description
search_term	The term that was searched for
result_count	The total number of results of the current page
more_results_available	Flag indicating if there are more result pages ("true" when more results are available, "false" otherwise)
total_results	The total number of results (when one page is returned, then this is the same as **result_count**)
total_pages	The total number of pages with results.
control	The fully built up HTML content of all results
container_results_style_class	The style class for the results container
individual_result_style_class	The style class for the individual results
plugin_error_message	Feedback from the plugin when an error occurred

For the individual result template:

Field for result template	Description
search_term	The term that was searched for
title	The title of the search result item
link	The URI to the found search result item
index	The index of the search result item (for full list of results)
basic_onclick	The basic on click function; to be used to update the hidden var that is necessary to allow a search result to be processed
custom_onclick	The optionally configured custom on click handler (configuration option **Custom onclick**, available when **Use result URL as link** is not checked/enabled)
submit_onclick	The standard submit function to trigger a post of the page
snippet	The short abbreviated and search term highlighted text
result_as_json	The full result object in JSON format
individual_result_style_class	The optionally configured result style class

When no results are returned:

Field for container template	Description
search_term	The term that was searched for
result_count	The total number of results
control	The fully built up HTML content of all results
no_results_available_message	The optionally configured custom message to indicate that no results were found
container_no_results_style_class	The 'no results' style class for the results container
individual_no_result_style_class	The 'no results' style class for the individual results

The individual results template is not applicable / relevant when there are no results returned.

The previous fields are the generic fields that the external search part will fill in. But each plug-in has its own configuration items and could potentially provide more information. All fields that are found by the plug-in are directly passed to the external search result and all configuration elements are in the sub structure **params**.

The specific parameters for the WordPress plug-in are:

WordPress specific params	Description
params.scheme	The scheme used to call the WordPress site (HTTP:// or HTTPS://)
params.port	The specific port number used to call the WordPress site
params.endpoint	The name of the endpoint, for example: posts
params.path	The actual endpoint used, for example: /wp-json/wp/v2/posts
params.order	The sort direction (asc or desc)
params.orderby	The field on which the results are sorted
params.max_results	The maximum amount of search results per returned page
params.page	The current page number

For the WordPress plug-in, the additional fields returned for each result item are:

WordPress specific result fields	Description
id	The WordPress internal ID of the item
element_type	The type of item (Post, Category, Tag, etc.)
creation_date	The timestamp the item was created (server local time)
creation_date_utc	The timestamp the item was created (UTC)
modified_date	The timestamp when the item was last modified (server local time)
modified_date_utc	The timestamp when the item was last modified (UTC)
excerpt	The excerpt of the result
content	The full content of the result
status	The result status (publish, future, draft, pending, private)
author	The author ID
tags_as_text	The comma separated list of tag IDs
categories_as_text	The comma separated list of category IDs

For the Wikipedia plug-in, the input parameters are:

Wikipedia specific params	Description
params.scheme	The scheme used to call the WordPress site (HTTP:// or HTTPS://)
params.port	The specific port number used to call the WordPress site
params.path	The actual endpoint used, for example: /wp-json/wp/v2/posts
params.order_by	The field and order on which the results are sorted
params.max_results	The maximum amount of search results per returned page
params.page	The current page number
params.offset	The result number to start retrieving the following results from. Calculated by doing **page * max_results**.

Additional result fields for Wikipedia plug-in are:

Wikipedia specific result fields	Description
id	The WordPress internal ID of the item
modified_date	The timestamp when the item was last modified (server local time)
content	The snippet of the search result (full content is not passed as that can be a very large dataset)
snippet	The snippet of the Wikipedia page
size	The total size of the found Wikipedia page
wordcount	The number of words in the Wikipedia page

Furthermore, date stamps can be converted directly with the template using the datetime function inside the interpolation, for example:

```
<div class="test_creation_date_utc"> Creation Date UTC:
$(datetime(creation_date_utc, 'date_time')) </div>
```

Supported input date formats are ISO format (YYYY-MM-DDTHH:mm:SS.fff+/-hh:mm) that supports timezones or regular format (YYYY-MM-DD HH:mm:SS, implicit assuming UTC) or LogicNets time number (YYYYMMDD.HHmmSS).

Examples

Currently there are two implementations for the external search: Wikipedia and wordpress.

Note that Google has a well defined API, but requires an application key to allow external apps to use google and is not currently supported in LogicNets external search.

Wordpress search limitations:

The current implementation is aimed at searching posts. Searching pages for example has slight different results and filter/search options. Additionally, authors, tags, categories etc could be exploited in future to make the search more specific / controlled.

Wikipedia search limitations:

Only basic search is implemented for wikipedia based sites where the content retrieved is the snippet.