



Aricie.LuceneSearch Documentation

Latest update: February 2013
Module version: 1.7.1



Table of contents

[1 About the module](#)

[2 Step-by-step installation](#)

[2.1 Prerequisites](#)

[2.2 Install the module](#)

[2.2.1 Install latest Aricie libraries \(required\)](#)

[2.2.2 Install the LuceneSearch PA](#)

[3 QuickStart](#)

[4 Concepts](#)

[4.1 Lucene](#)

[4.2 Native search items](#)

[4.3 Providers](#)

[4.4 Search vs filtering](#)

[4.4.1 Url parameters](#)

[4.5 Skin object and integration in skins](#)

[4.5.1 Installation](#)

[4.5.2 Configuration](#)

[5 Configuration](#)

[5.1 Portal settings](#)

[5.1.1 Indexing tab](#)

[5.1.2 Existing index tab](#)

[5.1.3 Search behavior tab](#)

[5.1.4 Autocomplete tab](#)

[5.1.5 Sitemap settings tab](#)

[5.1.6 Logs tab](#)

[5.1.7 Commands](#)

[5.2 General settings](#)



[5.2.1 Search layout tab](#)

[5.2.2 Search behavior tab](#)

[5.2.3 Results layout tab](#)

[5.2.4 Results behavior tab](#)

[5.2.5 Results template tab](#)

[5.2.6 Results highlights tab](#)

[5.2.7 Commands](#)

6 How to

[6.1 Search the content of doc and pdf documents?](#)

[6.2 Use a custom control for filters](#)

[6.3 Use a custom control for results?](#)

[6.4 Tweak the returned results?](#)

[6.5 Use the tokens available in the results?](#)

[6.5.1 Content related tokens](#)

[6.5.2 DotNetNuke related tokens](#)

[6.5.3 Search related tokens](#)

7 Support

1. About the module

Aricie – LuceneSearch (LS) is a search module for DotNetNuke that replaces the native search module and provides much more advanced features, flexibility and power to your website. Based on the ubiquitous Lucene engine, the module plugs directly into the DotNetNuke architecture for all your search needs. It goes beyond the default search engine bundled with DotNetNuke and also sidesteps some of the limitations you may encounter in the DotNetNuke ecosystem by letting you create search providers for modules that don't support DNN search natively, customizing how your search results are displayed, storing unlimited fields for your content, etc.

When you install LS, it comes bundled with a list of providers for commonly used modules and is ready to search your indexed data. From there, you can define new data to search as well as the way you want to search and display this data.

Required DNN version: 4.8.1 or above.

This documentation was created based on DNN 6.1 and the 1.7.1 version of LuceneSearch.

2. Step-by-step installation

1. Prerequisites

To install the module on your DotNetNuke site, you need to have a **DotNetNuke installation at least version 4.8.1**. Please note that LuceneSearch uses features from the **.Net framework 3.5**.

2. Install the module

To install LS on your website, please follow these steps

1. Install Latest Aricie libraries (required)

You must **install the Aricie.Shared common module**, which is available on <http://www.aricie.com>. In order to install it, you can follow the same instructions as the ones to install the LuceneSearch PA.

2. Install the LuceneSearch PA

1. Download the **Aricie LuceneSearch** module in the Download section of our website, www.aricie.com
2. Connect to your DotNetNuke site as Host
3. Go to **Host > Extensions**
4. Click **Install Extension Wizard**
5. Click on the Browse button and select the **Aricie.LuceneSearch.x.x.x.Release.zip** file
6. Click **Next**
7. You may receive a message warning you about file types that may not be accepted by your website. This is due to some of the files in the LuceneSearch package such as webservises, synonyms files, etc. Please check the check box to let the package install correctly and then click **Next**.

Upload Results

☒ If you have reached this page it is because the installer needs to gather some more information, before proceeding.

The installer maintains a list of allowable files (whitelist) that can be installed into your site. If you are confident that the package you are installing is from a reliable source then you can elect to ignore this list. Ignore File Restrictions? ☐

Next Cancel

8. On the package information display click **Next**.
9. The release notes will contain any important information regarding a new version. You don't have to install incremental upgrades, you can go straight with the version you want to install. Click **Next**.
10. Next is the licence agreement. Check the box and click **Next**.

11. Once the module is successfully installed, click **Return**

Info Installation successful. - Aricie.LuceneSearchResults

Info Deleted temporary install folder

EndJob Installation successful.



Return

- That's it! Now you are ready to add your module to a page.

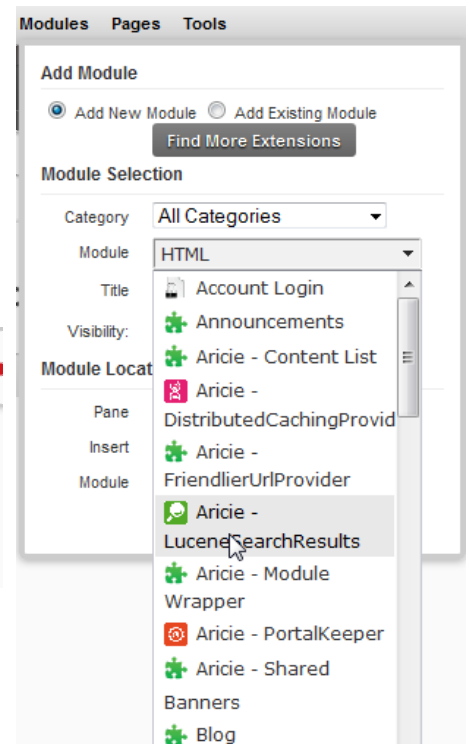
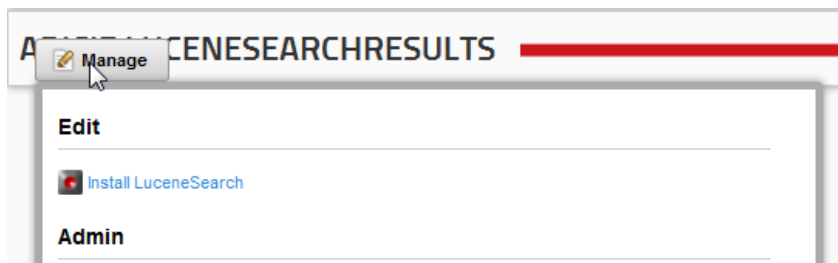
IMPORTANT: Some errors may appear in red during installation. If such an error happens, please check first if it is not a dependency error (in which case you may want to **check if you installed the Aricie.Shared module**). If you did, then don't hesitate to take a screenshot and send it to support@aricie.fr for us to have a look at it.

3. QuickStart

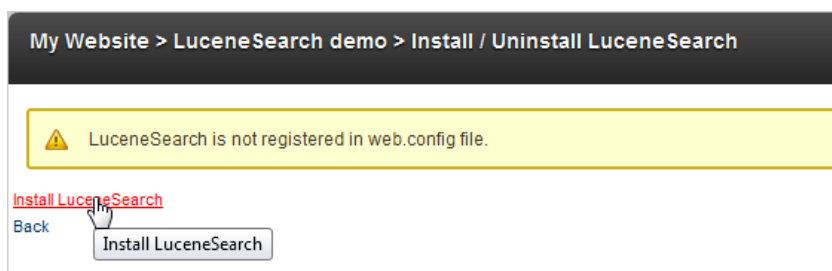
To install LS, please follow the standard DotNetNuke installation procedure. Below is a quickstart setup procedure using DotNetNuke 6.1. The process should remain the same for all supported versions.

Add the module to the page where you want to use LuceneSearch:

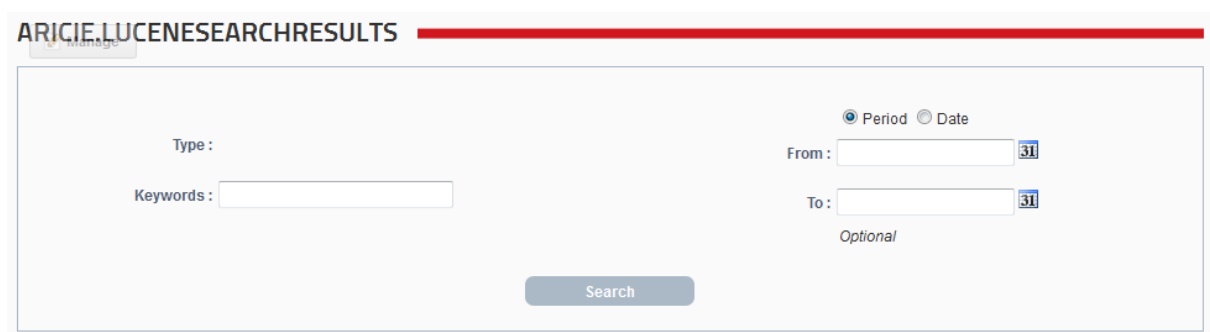
Once the module is installed you will have to register it into the DotNetNuke architecture. To do so, you must click on the Install LuceneSearch action command that appears in the module actions, under the Manage button for the module.



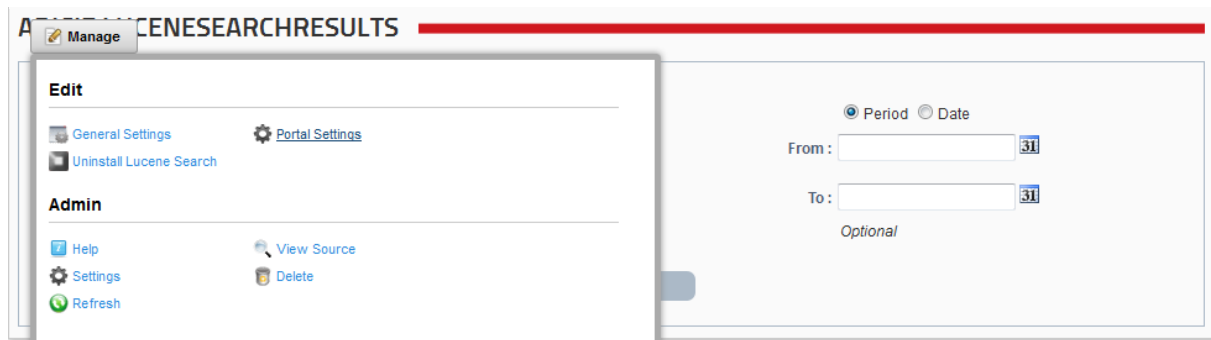
The module warns you that it isn't installed correctly in the web.config file. Click on the Install LuceneSearch link to confirm the installation



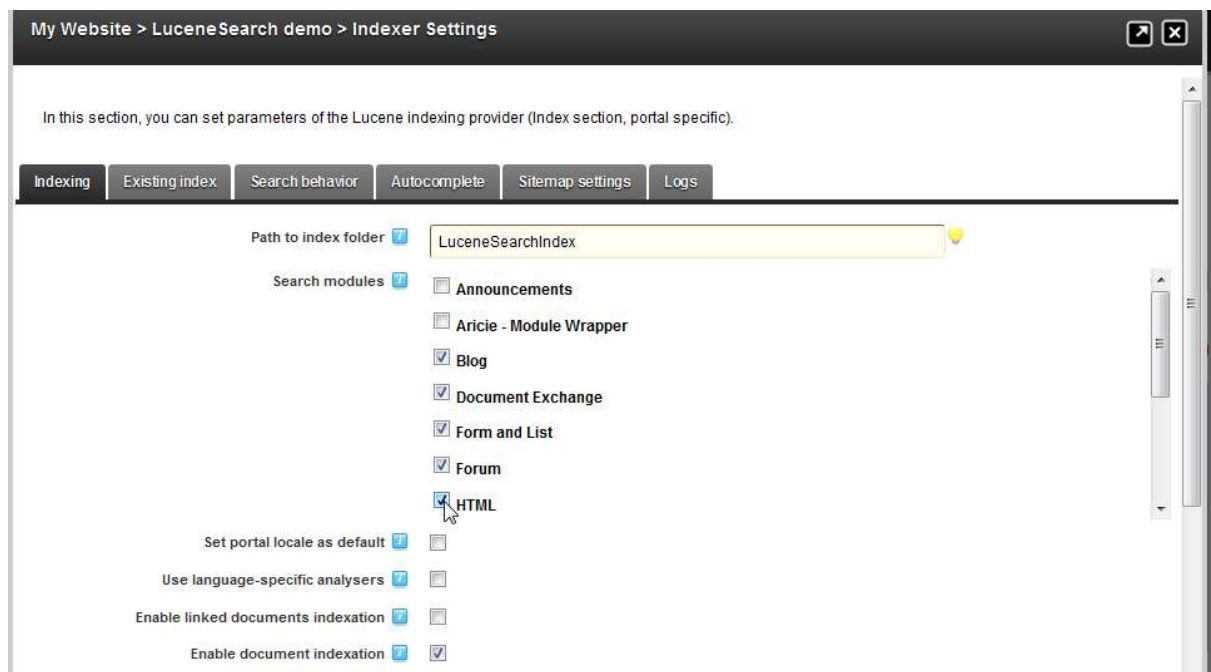
Now the default interface of LuceneSearch shows up.



No type is available because **no data has been indexed yet**. To index data you must configure LuceneSearch to tell it what data it needs to process. Go to the “portal settings” action in the manage menu



In the Indexing tab, select the modules whose content must be handled by LuceneSearch.



We are going to finish our quickstart by forcing DotNetNuke to re-index its data; this way LuceneSearch will get to process the data needed and populate its index for the first time.

Go to **Host > Search Admin**, and there click the “Re-index content” link. After a while, the indexing will be done and you LuceneSearch module is ready to display the data it found.

SEARCH ADMIN

Maximum Word Length:

Minimum Word Length:

Include Common Words: ☐

Include Numbers: ☒

[Re-Index Content](#)

Get back to your LuceneSearch module. The Type filter is now visible in the interface and provides the type of contents to search. Enter 'cycles' (DNN 6.1 default template is for a cycle store, so there will be hits for this term) and see what comes out of your first search.

ARICIE.LUCENESEARCHRESULTS

Type:

Keywords:

☒ Period ☐ Date

From:

To:

Learn About Awesome Cycles

Title:Learn About Awesome
Cycles
.....
HTML from Unspecified author,
published 25 October 2011, indexed 25 February 2013
http://localhost/dnn6/Default.aspx?tabid=60&language=en-US

That's it, we have our first result and you completed the setup for LuceneSearch. You can now start using its features which are further detailed.



4. Concepts

1. Lucene library

From Wikipedia:

Apache Lucene is a [free/open source information retrieval software library](#), originally created in [Java](#) by [Doug Cutting](#). It is supported by the [Apache Software Foundation](#) and is released under the [Apache Software License](#).

Lucene is based on a field concept, where a field and its content are added to the **Lucene index** and queries can be run against these fields. There is no predefined structure to the Lucene index, and different information and fields can be stored.

LuceneSearch takes advantage of this field concept by storing the original data coming out of the DotNetNuke search engine indexation and adding its own data to the structure it is given.

2. Native search items

In order to be searchable, a module must implement `ISearchable` from the DotNetNuke framework. This interface allows it to return search items that will be then passed by DotNetNuke onto the search engine that is configured.

Since LuceneSearch follows the conventions of the DotNetNuke search system, **any module that declares itself searchable will be handled automatically by LuceneSearch.**

3. Providers

What's more, it is also possible to replace the behavior of a searchable module through LuceneSearch custom providers. These providers can complete and replace search behaviors for modules. They can even implement search behaviors for modules that don't allow native search.

A series of tutorials tells you how to create your own providers.

[Providers in LuceneSearch: part 1 – standing on the shoulders of giants](#)

[Providers in LuceneSearch: part 2 – pretenders and replacements](#)

[Providers in LuceneSearch: part 3 – going solo](#)

LuceneSearch comes with a suite of providers which will help you hit the ground running.

- AjaxFAQ
- DNNArticle
- DocumentExchange
- Document
- EngagePublish
- Forum
- LiveTabs
- NewsArticle (articles and latest articles)
- PackFlash
- PropertyAgent
- Repository
- UltraMedia Gallery

We are also available to code providers to any module you desire. Please contact us directly at support@aricie.fr for more information.

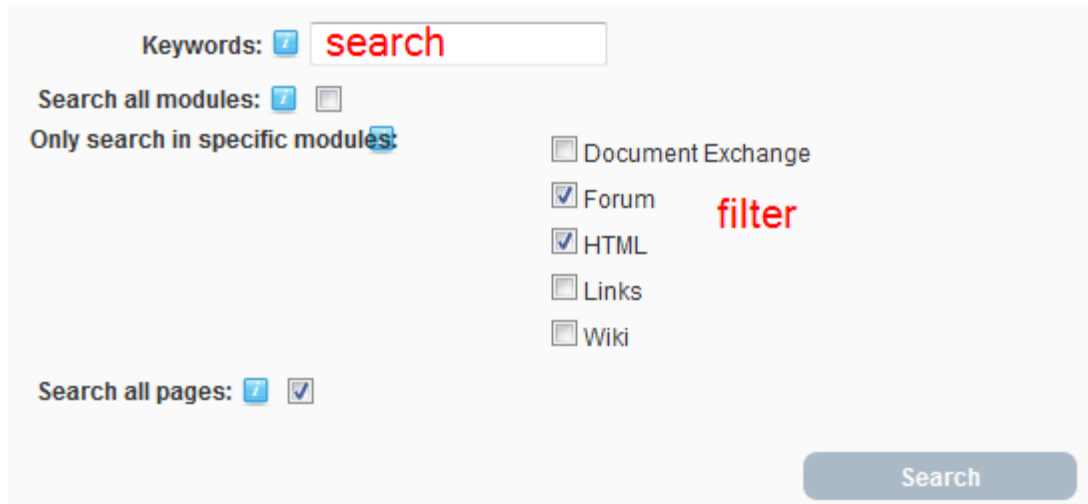
marque page TCH 28/02/13

4. Search vs filtering

There are two ways LuceneSearch can help you find content you need: search and filters.

Search will be run against the fields you configured either at the portal level or the module level (see **5.1.3 Search behavior tab** and **5.2.2 Search behavior tab**). What you type in a search element is directly taken into account by Lucene since it is taken into account as a string.

Filters are more declarative; they define at declaration time what fields they will query on and return a Lucene query directly.



Keywords:

Search all modules: ☐

Only search in specific modules:

- ☐ Document Exchange
- ☒ Forum **filter**
- ☒ HTML **filter**
- ☐ Links
- ☐ Wiki

Search all pages: ☒


In this example from the light filter search, keywords are a search field, and the module selection control as well as the page selection control are filter fields. This is because we know that they will have to be focused on one specific field in the LuceneSearch index.

1. Url parameters

Every time LuceneSearch refreshes the page, every parameter (search or filter) is passed in the url for the LuceneSearch module to retrieve them if possible. Such a refresh can occur when the current LuceneSearch module doesn't use Ajax, or if you configure it to redirect to another page.

In some cases, it is possible that the url contains search parameters that will not be usable by the destination LuceneSearch module. If you start from a LuceneSearch module that filters on the module name of your content and redirects to another LuceneSearch module without any filter displayed, the **destination module will pick up the url parameters to use them**. It will try to match its own filters and search controls to what was passed in the url and – failing – will apply them directly to the Lucene query. However, if you are connected as an administrator of the module, it will display a warning message.



 There are search informations from the url that have not been picked up by your current filters. This may be on purpose; if so you can disable this message by unchecking the "Enable warnings from url filtering" parameter in the general settings. If it is not on purpose, please check your settings to ensure that every search parameter is reflected on the results page.

This message is **only displayed if you are connected as an administrator and serves only as a warning** against incorrect configuration. **Users won't see it and it doesn't prevent the search from being launched.**

5. Skin object and integration in skins

LuceneSearch comes with a skin object that can replace the native search skin object from DotNetNuke. This skin objects handles the auto completion feature and search redirection to LuceneSearch if needed.

1. Installation

You have to declare the skin object [LUCENESEARCH] in your skin.

In the menu Host/Extensions, click “Create New Extension”, declare a new Skin object called LUCENESEARCH pointing to the SearchSkinObject.ascx control in LuceneSearch.

Go into the skin used for the portal. Replace the DNN token [SEARCH] by the new token [LUCENESEARCH].

2. Configuration

It is possible to configure the skin object to behave according to what is needed by your skin.

- **ResultsTabId**: lets you specify the page you want the skin object launch the search on.
- **UseLanguageFilter**: automatically filter the query with the current locale.
- **Submit**: submit text
- **CssClass**: define the css class on the command button
- **ShowFilter**: display module filtering
- **DisableFuzzyRatio**: disables the fuzzy factor in the search query
- **SubmitImageUrl**: url for the image button
- **IncludeSkinObjectCSS**: lets the skin object include its css. If true, the SkinObjectCSS parameter is used
- **SkinObjectCSS**: the path to the CSS to include if necessary. By default, the included css is the same one as shown on <http://www.aricie.com>
- **LocalResourceFile**: resource file used for translations

5. Configuration

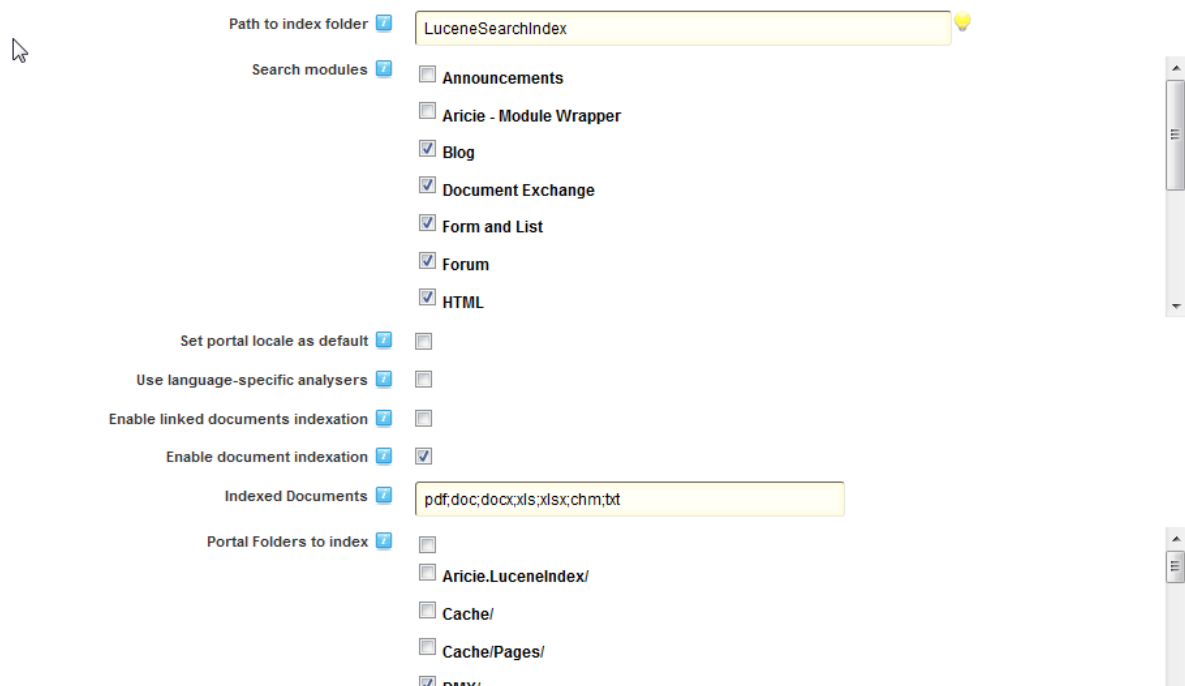
1. Portal settings

LuceneSearch works independently on each portal in a website. So the portal settings are shared between all LuceneSearch modules in this portal. Here is a look at what is available for configuration



1. Indexing tab

The indexing tab drives what data will be indexed and how.



The screenshot shows the 'Indexing' tab configuration interface. It includes the following settings:

- Path to index folder:** A text field containing 'LuceneSearchIndex'.
- Search modules:** A list of checkboxes for various modules:
 - ☐ Announcements
 - ☐ Aricie - Module Wrapper
 - ☒ Blog
 - ☒ Document Exchange
 - ☒ Form and List
 - ☒ Forum
 - ☒ HTML
- Set portal locale as default:** ☐
- Use language-specific analysers:** ☐
- Enable linked documents indexation:** ☐
- Enable document indexation:** ☒
- Indexed Documents:** A text field containing 'pdf,doc;docx;xls;xlsx;chm;txt'.
- Portal Folders to index:** A list of checkboxes for various folders:
 - ☐ Aricie.LuceneIndex/
 - ☐ Cache/
 - ☐ Cache/Pages/
 - ☒ nmx/

Path to index folder: for each portal a Lucene index must be created. This is the name of the folder where the Lucene index will live.

Search modules: this list contains everything that can be indexed by the LuceneSearch module, whether it is a native module or a provider that adds data or replace the module behavior.

Set portal locale as default: LuceneSearch handles locales during its indexation if they are available. However locales are not supported by the native DotNetNuke search system and native modules won't be able to register the culture they are related with. Checking this setting will

automatically add the current portal's default language to the indexed data.

Use language-specific analysers: analysers in Lucene are tools that will try to determine whether two words are supposed to be the same: "indexed" and "indexing" are two different words yet they share a common root. If checked, LuceneSearch will only keep the root data for these two words and will match them both to a search for "indexation"

- ---

Important: you should clear the index or wait until it is naturally updated, since the tokenizer used at query time must match the tokenizer used at index time for accurate results.

Enable content taxonomy indexing: This setting is only visible if the standalone taxonomy provider is enabled in the configuration of LuceneSearch. If checked, special items will be created as taxonomy documents. These taxonomy documents will try and redirect you to the search interface

Enable linked documents indexing	<input checked="" type="checkbox"/>
Reflexive indexation: accepted file extensions	pdf,doc;docx,xls;xlsx;chm;txt
Reflexive indexation: maximum operation time (seconds)	30

Enable linked documents indexation: This setting is only visible if the reflexive provider is enabled in the configuration of LuceneSearch. If checked, the LuceneSearch engine looks for documents in the content that has already been indexed. These documents' indexation will be configured by the following two parameters

Reflexive indexation: accepted file extensions: lists the document extensions (separated by semi-columns) that the reflexive provider will look for.

Reflexive indexation: maximum operation time (seconds): sets the maximum time to spend indexing the documents before continuing. Since documents can take a long time to index, LuceneSearch acts in incremental steps. This setting allows the indexation to stop after the elapsed time and resume in the next indexation.

Enable document indexation ☒

Indexed Documents

Portal Folders to index ☐

- ☐ Aricie.LuceneIndex/
- ☐ Cache/
- ☐ Cache/Pages/
- ☒ DMX/
- ☒ DMX/2012/
- ☒ DMX/2012/04/

Enable document indexation: if checked, the LuceneSearch engine will look for documents in the portal folders on the server disk. These documents' indexation will be configured by the following two parameters. Please see **6.1 Search the content of doc and pdf documents**.

Indexed Documents: lists the document extensions (separated by semi-columns) to be indexed.

Portal Folders to index: a checkbox list of the folders that exist under the portal folder on the server disk. Each folder will be searched for documents with an extension in the Indexed Documents parameter.

2. Existing index tab

Indexing
Existing index
Search behavior
Autocomplete
Sitemap settings
Logs

Upgrading duration

Indexing duration

Max nb of documents per commit

☒ Max indexed age

Formatted duration

☒ Details

Min nb of documents to turn over

Old documents indexing ratio

Trust publication Dates ☐

The existing index settings handle how LuceneSearch will handle the creation and old items management.

Upgrading duration: lets you define how long in seconds the upgrade process of LuceneSearch will



last. The upgrade step usually is the longest operations in the indexation, such as parsing and indexing an external document, or making use of a very big datasource.

Indexing duration: lets you define how long in seconds the storage process of LuceneSearch will last.

Max nb documents per commit: The maximum number of documents that LuceneSearch will store in one go. This parameter can be tweaked for optimized performance purposes. Since performance depends on the documents type and size, it should be configured depending on your data.

Max indexed age: Sets the age after which a document is declared as obsolete.


Min nb of documents to turn over: the minimum number of documents that will be refreshed during each indexation.


Old documents indexing ratio: the ratio of documents that will be refreshed during each indexation.

Trust publication dates: LuceneSearch tries to optimize the time spent indexing by checking if the publication date of an item is more recent than the latest indexation. However some modules don't fill in the publication date correctly when returning the data natively, and this setting lets you ignore the publication date returned.


3. [Search behavior tab](#)

Indexing
Existing index
Search behavior
Autocomplete
Sitemap settings
Logs


Use shared index reader ?  ☒

Desktop filter values 

☐ Blog
☒ Document Exchange
☐ Form and List
☒ Forum
☒ HTML
☒ Links
☒ Wiki

Search fields 

☒ Title
☒ Description
☒ Content
☒ Published Date
☒ Page Name
☐ Page Title
☐ Page Description

Synonym Map Path 

The search behavior regroups the configuration of how the data will be searched.

Use shared index reader: a common index reader is shared and kept open across the various threads and functions querying the index. Otherwise, a specific index reader is created and closed on each operation.

Desktop filter values: this list lets you define what modules will appear in some filter menus in the UI.

Search fields: you can define the fields that will be searched by default on the whole portal. You will be able to override these values at the module level.

4. [Autocomplete tab](#)

Manages autocomplete settings

Enable autocomplete ☒

Server settings

Min size for autocomplete

Number of search results

Fields returned

☒ Description

☒ Description

☒ Description

☒ Description

Add field to return:

Sorting fields

AuthorName ↑

TabTitle ↓

AuthorName

Add field to sort:

Client settings

The autocomplete adds an autocomplete behavior to the LuceneSearch skin object in order to allow fast search from anywhere on the skins.

Enable autocomplete: check to enable autocomplete feature.

1. *Server settings*

Server settings

Min size for autocomplete

Number of search results

Fields returned

☒ Description

☒ Description

☒ Description

☒ Description

☒ Description

Add field to return:

Sorting fields

AuthorName ↑

TabTitle ↓

AuthorName

Add field to sort:

Min size for autocomplete: the autocomplete will be triggered after this number of characters has

been typed in the skin object textbox


Number of search results: the autocomplete will return at most this number of results.

Fields returned: you can configure what fields will be returned by the server to your client-side object. This is so you won't send on the wire unused data.

Sorting fields: you can define what fields the returned values will be sorted against.

2. *Client settings*


Enable custom positioning  ☒

Custom positioning of autocomplete 

```
{ "my": "left top", "at": "left bottom", "collision": "none" }
```

Enable custom positioning: when checked, lets you define how the autocomplete will be positioned regarding the skin object.

Custom positioning of autocomplete: lets you type how the autocomplete will appear in regard to the skin object.

Enable header text  ☒

Header text 

```
<div style="color: silver;">Suggestions de recherche</div>
```

Enable header text: check this to define a custom html header that will appear on top of your results list.

Header text: the html header that will appear on top of your results list.

Enable grouping ☒

Group name

Group template text

Enable grouping: check to group items.

Group name: defines what field the items will be grouped against; when grouping, the sort order takes the upper hand against sorting. The results will be sorted, then grouped which can alter the previous sorting.

Group template text: the javascript template that needs to be applied to the group. The template uses the micro-templating engine which can be found here:

<http://www.west-wind.com/weblog/posts/2008/Oct/13/Client-Templating-with-jQuery>

Item template text

No results text

Enable highlights ☐

Handling method for click on elements

Include custom styles and scripts ☒

Url to a version of jQuery

Url to a version of jQuery.UI

Overwrite existing versions of jQueryUI ☐

Url to your Autocomplete custom CSS

Item template text: the javascript template that needs to be applied to each item. The template uses the micro-templating engine which can be found here:

<http://www.west-wind.com/weblog/posts/2008/Oct/13/Client-Templating-with-jQuery>

No results text: what will appear when no results are found.

Enable highlights: if checked, the autocomplete will try and display the searched terms in highlights in the autocomplete dropdown.

Handling method for click on elements: if you want to open some links in different windows, for example external websites, you can write down a javascript function here that will be run against the item clicked on. If the function returns null, the link is opened in the current window. If the function returns a string, a new window will be opened with this identifier. You can return the same value for different item types so that they all open in a different window.

Include custom styles and scripts: LuceneSearch's autocomplete tries to display itself as robustly as possible, but there may be times when you don't want it to include scripts or styles directly in your code. In order to avoid that, you can check this setting to control what will be included in the webpages of your site.

Url to a version of jQuery: type the url to the jQuery version needed

Url to a version of jQueryUI: type the url to the jQueryUI version needed

Overwrite existing versions of jQueryUI: if checked, the autocomplete will try to replace previous versions of jQueryUI (for example if some modules pre-embed them in their resources).

Url to your Autocomplete custom CSS: the url to the css you want to use with the skinobject to skin the autocomplete.

5. Sitemap settings tab

Default Priority Percentage	<input type="text" value="70"/>
Compute Priority	<input checked="" type="checkbox"/>
Compute Change Frequency	<input checked="" type="checkbox"/>

Because Lucene is able to index items beyond the page level by indexing modules' content, it is also able to create a much more comprehensive site map for your website SEO.

Default Priority Percentage: set the percentage of priority by default, the value range between 0 and 100.

Compute Priority: enable to compute the priority automatically.

Compute Change Frequency: enable to compute the change frequency automatically.

6. Logs tab

Trace Indexing in Event Log ☐ ☒

Activate search logs ☐ ☒

Logs duration ☐

Trace indexing in event log: check to generate event progressing logs in the admin event viewer. This allows monitoring the metrics of the indexing procedure according to the various parameters.

Activate search logs: traces the searches launched by the users.

Logs duration: number of days to retain logs.

7. Commands

The following commands are available on the portal settings:


- Save and continue: saves the current settings and stays on the settings edition page
- Save and return: saves the current settings and goes back to the LuceneSearch module
- Cancel: goes back to the LuceneSearch module without saving
- Clear portal index: clears the index for the portal


2. General settings


Each instance of a LuceneSearch module can be configured independently to allow users to finetune what behaviors are desired.





1. Search layout tab


Display search panel  ☒

Display searching filters  ☒

Select searching filters  Standard filter ▼

Display Search button  ☒

Display Reset button  ☒

Display search panel collapse/expand  ☒

This tab contains the control dealing with the presentation of the search area for the module.

Display search panel: displays or hides the search panel. Unchecking this parameter will hide all others parameters.

Display searching filters: displays or hides the searching filters.

Select searching filters: only displayed if the “Display searching filters” is checked. This dropdownlist lets you select alternate search interfaces.


Filters control path: only displayed if the “Select searching filters” parameter is set to “Custom filter”. This lets you type the path to a user control (*.ascx) completely customized to your needs. Please see [6.2 Use a custom control for filters](#) for more information.

Display search button: displays or hides the search button, independently of the searching filters.

Display reset button: displays or hides the reset button, which empties the user interface.

Display search panel collapse/expand: displays or hides accordions in the UI.

2. Search behavior tab

Activate Ajax	<input checked="" type="checkbox"/>
Enable warnings from url filtering	<input checked="" type="checkbox"/>
Apply Default Language Filter	<input type="checkbox"/>
Additional Lucene Filter	<div style="border: 1px solid #ccc; height: 60px; width: 100%;"></div>
Enable Token Replace on Additionnal Filter	<input type="checkbox"/>
Permission filter	Filter on query ▼
Search fields	<input type="checkbox"/> Title <input type="checkbox"/> Description <input type="checkbox"/> Content <input type="checkbox"/> Published Date <input type="checkbox"/> Page Name <input type="checkbox"/> Page Title <input type="checkbox"/> Page Description
Default fuzzy value	100 

The search behavior tab contains every setting that handles the way queries are processed in the module.

Activate Ajax: enables the module to search data in an asynchronous way like Ajax behavior. If you check it, the results will appear without page refresh when you click the button “Search”. If you disable it, you will have to select the page you wish to display after the search has been launched.

Results page Id: This parameter only appears if you unchecked the activate Ajax parameter. It’s a dropdownlist of your portal pages in which you can choose what page the search is going to redirect to. If you select the empty entry, the search will simply postback to the server but stay on the same page.

Enable warnings from url filtering: when checked, this settings displays a message if the search interface wasn’t able to process every parameter that was optionally passed in the url. This warning is only displayed to administrators and is useful during the initial configuration setup.

Apply default language filter: check to apply the current user language as a language filter. Make sure indexed contents contain language information, and set the indexer settings so that no localized contents have the portal language by default.

Additional lucene filter: lets you add a Lucene query directly from the configuration in order to finetune the search you want. To see what the syntax looks like you can consult the page at http://lucene.apache.org/core/old_versioned_docs/versions/2_9_1/queryparsersyntax.html

Enable token replace on additional filter: lets you use tokens in the lucene filter so you can use dynamic values at search time. Available tokens are the common DotNetNuke tokens.

Permission filter: A dropdownlist that lets you select between filtering the results according to the user rights or not filtering the users. Credentials for the user at the time of search are compared to the credentials of each stored document (according to the page and module settings, and even the document settings for some modules) and only matching results are returned. The document credentials are stored during indexation.

Search fields: the list of fields that will be searched for search elements on this module. This setting overrides the search fields from the portal settings.

Default fuzzy value: default value for fuzzy configuration on the search controls.

3. Results layout tab

Show Results panel	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Show Backlink	<input type="checkbox"/>	<input type="checkbox"/>
Display result panel collapse/expand	<input type="checkbox"/>	<input type="checkbox"/>
Display # of results	<input type="checkbox"/>	<input type="checkbox"/>
Enable Paging	<input type="checkbox"/>	<input type="checkbox"/>
Limit result number	<input type="checkbox"/>	<input type="checkbox"/>
Custom grouping	<input type="checkbox"/>	<input type="checkbox"/>
Custom sorting	<input type="checkbox"/>	<input type="checkbox"/>
Show print button	<input type="checkbox"/>	<input type="checkbox"/>

The result layout settings control how the results will be displayed to your users when search results are returned.

Show results panel: hides or display the result panel. This is useful if the search panel and result panel are on different pages.

Show Backlink  ☒

Backlink layout 


```
<div class="LS_BackLink"><a href="[BACK_LINK_URL]" >[BACK_LINK_LIB]
</a></div>
```

Show backlink: displays a backlink on LS details pages


Backlink layout: token-enabled html snippet that will be used for the backlink from the details page

Display result panel collapse/expand: displays or hides accordions in the UI.


Display # of results: displays the total number of results found by the search query.

Enable Paging  ☒

Display # of pages  ☒

Pager navigation  Numeric + First/Last

Pager size  5 

Custom paging  ☒

Enable paging: activates the paging in the results UI.

Display # of pages: displays the total number of pages found by the search query.


Pager navigation: selects the style of pager that will be used by the results panel.


Page size: sets the number of items returned per page.


Custom paging: lets users define their own page size through a dropdown list on the UI.

Custom grouping: lets users group the results on some interfaces.

Custom sorting: lets the users sort the results on some interfaces.

Show print button  ☒

Disable skin for printing  ☒

Disable pagin for printing  ☒

Show print button: displays or hides a link to a print page for the search results


Disable skin for printing: check to have the print window stripped of its skin.


Disable paging for printing: for some report-type printing, paging is not desirable, the user needing the whole data set to print. Check to retrieve all data on one page.

4. Results behavior tab


Show no results until Search  ☒

Redirect if single result  ☐

Group by : 

Descending group order  ☐

Sort by : 

Descending sort order  ☐

Cache duration  

The results behavior tab drives how the results will appear on your module.

Show no results until Search: if checked results will only be displayed after a search is launched by the user. If unchecked, a search is launched as soon as a user arrives on the module.

Redirect if single result: if only one result is found, the module automatically redirects the user to the result link.

Group by: the field you want to group against by default


Descending group order: reverse the grouping order

Sort by: the field you want to sort against by default

Descending sort order: reverse the sort order

Cache duration: the duration during which the results are cached

5. Results template tab

Template configuration 


Display mode : TemplateView

Scope : ☐ Host ☐ Portal ☒ Module

☐ Customize templates for each locale English (United States)


☐ Customize templates for each content

Result templates



```
<div class="item" align="left">
  <h3>
    <a class="title" href="[Result.GoToUrl]">
      [Result.Title]</a></h3>
  <div class="content">
```

Detail templates



```
<div class="item" align="left">
  <h3>
    <span class="SubHead">[Result.Relevance]% </span><a class="title" href="[Result.GoToUrl]">
      [Result.Title]</a></h3>
  <div class="content">
```

The results template tab lets you define how your results will be displayed. It lets you define your templates at three levels:

Display mode: lets you choose between 3 display modes for the search results.

- **TemplateView:** displays an html template where you can use the token replace on the result item. You can use tokens on the result item: Relevance, GoToUrl, Title, Description, Highlights, FriendlyModuleName, AuthorName, PubDate, IndexedDate, GoToFullUrl. These are common tokens; to see the full list of available tokens please see **6.5 Use the tokens available in the results?**
- **GridView:** displays the grid template. The grid template uses the same tokens as the Template view.
- **CustomView:** the custom view lets you enter the path to a control which will be used as the result control. See section 6.2 for more information about how to use this custom results control.

Scope: lets you define how the templates you are working on will be handled. Host defines templates for the whole website, Portal for the portal and module only for the module you are working on.

Customize templates for each locale: lets you define different templates for different locales.

Customize templates for each content: lets you define different templates for different content types, usually different modules.

Results templates are used for search results, details templates are used for content that can manage a detailed view (at the time, very few contents are able to display details; usually details are used for content that cannot be displayed in the native DNN UI, such as documents).

Templates are looked for from the **most specific to the least specific**. A template declared in a module will take precedence over a template declared for a host. In the same way, a template declared for a locale is more important than a generic template. Templates per content override templates per locale which override simple templates; module templates override portal templates which override site templates.

6. Results highlights tab

Show highlight section	<input checked="" type="checkbox"/>
Max # of samples per result :	<input type="text" value="4"/>
Max size of a sample :	<input type="text" value="50"/>
Highlight section	<input type="text" value="[FieldName]: ...[Fragment]..."/>
Fragment separator	<input type="text" value="
"/>
Highlight layout	<input type="text" value="[Content]"/>

The results highlights allow you to put the information that was searched for in evidence by using the highlight mode of Lucene.

Show highlight section: enables highlights to be computed for display

Max # of samples per result: maximum numbers of highlights that will appear for each entry

Max size of a sample: the size of the surrounding region for the highlight

Highlight section: html template with tokens that will display the highlight section

Fragment separator: html template of what is inserted between each fragment

Highlight layout: html template with tokens that will display the highlight itself

7. **Commands**



The following commands are available on the module settings:

- Save and continue: saves the current settings and stays on the settings edition page
- Save and return: saves the current settings and goes back to the LuceneSearch module
- Cancel: goes back to the LuceneSearch module without saving

6. How to

1. Search the content of doc and pdf documents

To index Word and Excel files you must install “IFilter for Office” on your webserver:

<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=5cd4dcd7-d3e6-4970-875e-aba93459fbee>

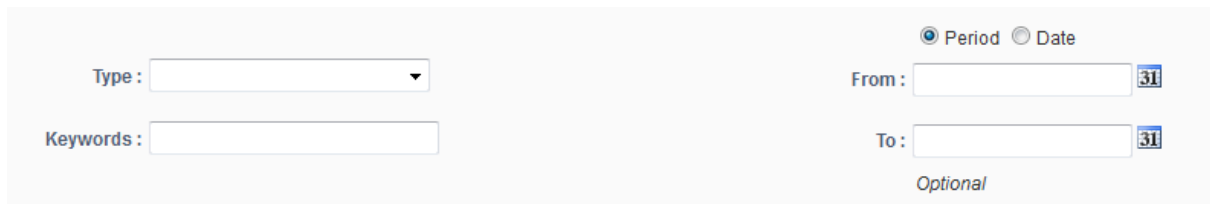
To index PDF files you must install “IFilter for PDF” on your webserver:

<http://www.adobe.com/support/downloads/detail.jsp?ftpID=2611>

2. Customize your search interface

If the default search controls bundled with LuceneSearch don’t fit with your website or if you need to create your own search interface, create a control that will be used as a search filter control.

To create such a control, the best thing is to start from the CommonFilter.ascx as an example.



The screenshot shows a search filter control with the following elements:

- Type:** A dropdown menu.
- Keywords:** A text input field.
- Period/Date:** Radio buttons to select between "Period" (selected) and "Date". Below these are "From:" and "To:" date pickers, both showing "31".
- Optional:** A label below the date pickers.

This above control shows three kinds of filters:

- **Type filter** is a dropdown list filter: it is attached to a Lucene field and pulls out all possible values for this field into a dropdown. Selecting an item in the dropdown list will then create a Lucene query that will match exactly the value you chose. This filter is best used on small lists in order to avoid creating a dropdown list with too many items
- **Keywords filter** is a prefix filter: any text typed in this area will be searched as if it were the beginning of a word. Typing “index” will return content with “indexed”, “indexing”, etc. Typing multiple words - separated with a space - will search on all the words without order, with only the last one used as a prefix. If no fields are defined for this filter, it defaults to using the search fields in the configuration.
- **Period filter** lets you search for a date or a period in the index. By default, it points to the published date. You can set it to any date field your content uses.

In order to use these filters in your custom control, just **register them from the LuceneSearch module as regular controls** in the \DesktopModules\Aricie.LuceneSearch\Controls folder. Use them in your markup. Here a very simple filter that only uses the prefix filter on the Description field.

```
<%@ Control Language="vb" AutoEventWireup="true" Inherits="Aricie.DNN.UI.Controls.AricieUserControlBase, Aricie.DNN" %>
<%@ Register Assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" Namespace="System.Web.UI" TagPrefix="asp" %>

<%@ Register Src="/DesktopModules/Aricie.LuceneSearch/Controls/PrefixFilter.ascx" TagPrefix="Filters"
    TagName="Prefix"%>

<div>
    <Filters:Prefix id="prefix" runat="server" QueryMode="Filter" Fields="Description" />
</div>
```

There are plenty of other filters available and you can also create your own. **If you need assistance to create your custom control filters**, please contact us at support@aricie.fr.

3. Use a custom control for results

Just as for the search controls, you can decide to use a custom control to display the LuceneSearch results.

First you must create a user control that inherits from the following class:

`Aricie.DNN.Modules.LuceneSearch.UI.LuceneResultsControllerBase`

Then in the Load event, you will have to call the `InitQuery` method from the base class when first loading your control:

```
Protected Overrides Sub OnLoad(e As System.EventArgs)
    If Not Page.IsPostBack Then
        InitQuery()
    End If
    MyBase.OnLoad(e)
End Sub
```

Once your control inherits from this class, you will have to override the method `DisplayResults`.

This method is called with a list of `LuceneSearchResultsInfo` that matches the query that was sent to Lucene. This list can then be used as a datasource for any databinding control you want in the user control.

```
Public Overrides Sub DisplayResults( _
    resultsList As System.Collections.Generic.IList(Of Aricie.DNN.Modules.LuceneSearch.Business.LuceneSearchResultsInfo), _
    nbResultFound As Integer, _
    sortFieldName As String, _
    printMode As Boolean, _
    Optional pageIndex As Integer = -1)
```


Although this looks simple, more options are available in order to drive configuration in the base class: grouping, paging, etc. Your control will have to pass data to the base class in order to transmit the configuration set by the user to the query engine of LuceneSearch.

As for every open behavior for LuceneSearch, we are available to help you if needed. Just contact us at support@aricie.fr.

4. Tweak the returned results

It is possible to tweak the results by using the parameter “Additional Lucene filter” in the “Search behavior” tab for the results configuration. **By typing the query that should be added to the search you can exclude some pages from being in the results, or some modules.**

For example, if you have a HTML module as a footer on every page, you can choose to exclude it from the results by using its title:

Additional Lucene Filter 

```
+ModuleTitle:*|-ModuleTitle:footer
```

This query lets you retrieve every module except the ones where the title is “footer”. Using this technique you can force only a subset of results to be returned.

To exclude several modules from the Lucene index, repeat the command as follows (mind the Lucene syntax. Spaces are important !).

```
+ModuleTitle:* -ModuleTitle:Title1 -ModuleTitle:Title2 -...
```



5. Use the tokens available in the results

Here is a list of tokens available within the results templates. The syntax to insert a token like Title in your template will be [Result:Title].

1. Content related tokens

- Title
- Description
- Author: author id in DNN
- PubDate
- SearchKey
- Guid
- AuthorName
- AdditionalFields: a list of data indexed by fields; this list can contain various data depending on the provider that handled the item
- ForcedUrl: optional url that the item must redirect to
- GoToUrl: computed url for the item
- LocaleCode
- Image: image id in DNN
- IconPath
- Category
- ContentTaxonomy: taxonomy (tags) from DNN, as a list of string. This field is reserved and not used at the time.
- ModuleTaxonomy: taxonomy of the module as a list of string if applicable
- TabTaxonomy: taxonomy of the tab as a list of string if applicable
- CompoundedTaxonomy: the merged list of all taxonomies
- Culture: .Net Culture info

2. DotNetNuke related tokens

- PortalId
- TabId
- ModuleId
- ModuleName
- ModuleInfo: as the DotNetNuke ModuleInfo class
- FriendlyModuleName
- ModuleTitle
- TabInfo: as the DotNetNuke TabInfo class
- TabName
- TabTitle



- TabDescription
- TabViewRoles: string containing the authorized roles for the tab
- ParentTab: as the DotNetNuke TabInfo class
- ParentTabId
- ParentTabName
- ParentTabTitle
- ParentTabDescription

3. **Search related tokens**

- Occurrences
- Relevance
- Score
- Boost
- Highlights: computed highlights if the settings authorize them
- Field
- ProviderName
- ItemType: translated item type for the index
- ItemGroup: translated item group in the index
- RawItemType: untranslated item type for the index
- RawItemGroup: untranslated item group in the index
- IndexedDate

7. **Support**

Our technical support team is committed to providing top support service. Please check out our support conditions on our website.

For best reactivity, please contact us on our Support forum:

<http://www.aricie.com/en/resources/support.aspx>

Need an special feature? Just tell us what you need and we will make a quote within 24 hours.

Have a great Aricie experience with Aricie.LuceneSearch!