gs.search.base Documentation

Release 2.3.2

GroupServer.org

February 01, 2016

Contents

CHAPTER 1

The core search interface for GroupServer

Author Michael JasonSmith

Contact Michael JasonSmith <mpj17@onlinegroups.net>

Date 2016-02-01

Organization GroupServer.org

Contents:

1.1 JavaScript

This product provides a JavaScript module as a Zope browser resource. Any Zope or Plone project should be able to use this product by placing the following line in a page template:

```
<script type="text/javascript"
        src="/++resource++gs-search-base-js-20160201.js"
        defer="true"> </script>
```

Users of other systems are invited to copy the file `gs/search/base/browser/javascript/search.js` out of this product.

A minified version of the script is available, too:

```
<script type="text/javascript"
        src="/++resource++gs-search-base-js-min-20160201.js"
        defer="true"> </script>
```

1.1.1 JavaScript API

The API is provided by the *`GSSearch()`* class.

class `GSSearch`(*widget*, *ajaxPage*, *offset*, *limit*, *additionalQuery*, *advancedSearchId*)

Arguments

- **`widget`** (*`string`*) – The selector for the HTML widget (normally an ID-selector).
- **`ajaxPage`** (*`string`*) – The page to query to get the AJAX results (see AJAX Page).

- **offset** (*int*) – The *initial* offset into the search.
- **limit** (*int*) – The number of results to show.
- **additionalQuery** (*object*) – Extra items to pass to the `ajaxPage` as part of the query. Set it to `{}` if there are none.
- **advancedSearchId** (*object*) – The *Advanced Search* link. This will be updated to reflect the current search.

load()
The *load()* method makes a query and load the results. The results are not loaded during the creation of the widget because in many circumstances (such as with the Posts [1] tab with GroupServer groups) the results do not need to be loaded when the widget is created.

results_shown()
The *results_shown()* method returns `true` if the results have been loaded, and `false` otherwise.

1.1.2 Behaviour

The JavaScript binds event handlers to the three buttons in the interface: *Search*, *Next*, and *Previous* (see HTML). Whenever these three buttons are pressed, or the *load* method is called, the following occurs:

1. The current results are hidden,
2. The loading message is shown, and
3. A `POST` request is made to the AJAX Page.

Once the request returns the loading message is hidden, the results are shown, and an *event* is raised.

Both the *Next* and *Previous* buttons modify an internal counter, that keeps track of the current *index* into the search-results, which is passed to the AJAX Page. It is always a positive number; if it is 0 the *Previous* button is disabled, while the *Next* button is disabled when the number of search-results is less than the `limit` that is set during creation.

There are **two** cases of no results.

1. The user searches for something, but nothing matched the search. In this case the HTML with the `gs-search-failed` class will be shown.
2. There is nothing to search. In this case the HTML marked with the `gs-search-empty` class will be shown, and the search-entry will be hidden. It is good practice to *mute* this HTML, because this is not an error state.

The system determines the difference between the two cases by looking at the search-entry: if it has text and the AJAX Page returns nothing then it must be the first case; else it the second.

1.1.3 Event

After the search-results have been loaded the search-widget will trigger a `resultsloaded` event. External systems may `bind` to this event to add functionality. For convenience [2] a constant for this string, `RESULTS_LOADED_EVENT`, is exported by the class.

[1] See `gs.group.messages.posts` <https://github.com/groupserver/gs.group.messages.posts/>
[2] Convenience, and the fact that I prefer constants to strings.

1.2 HTML

The various subsystems that wish to support the *search widget* must product HTML that conforms to the following structure:

- Search widget: `.gs-search`
  - Text-entry: `.gs-search-entry`
    - Search entry: `input`
    - Button: `button`
  - Loading: `.gs-search-loading`
  - Results: `.gs-search-results`
  - No results found: `.gs-search-failed`
  - Nothing to search: `.gs-search-empty`
  - Toolbar: `.gs-search-toolbar`
    - Next: `.gs-search-toolbar-next`
    - Previous: `.gs-search-toolbar-previous`

Below is a typical layout for a search widget. In addition to the classes above, some classes used by Bootstrap are shown, as well as WAI-ARIA roles. Neither is necessary, but both work with the search widget:

```
<div class="gs-search">
  <div class="gs-search-entry search input-append">
    <input type="search" name="s" placeholder="Search&#8230;"
           autocomplete="on" value="" title="Search"/>
    <button id="gs-group-messages-topics-search-button"
            class="btn">Search</button>
  </div><!--gs-search-entry-->
  <p class="gs-search-loading" role="status">
    <span data-icon="&#xe619;" aria-hidden="true" class="loading"> </span>
    Loading&#8230;
  </p><!--gs-search-loading-->
  <div class="gs-search-results">
     
  </div><!--gs-search-results-->
  <p class="gs-search-failed">
    No topics were found.
  </p><!--gs-search-failed-->
  <p class="gs-search-empty muted">
    There are no topics in this group.
  </p><!--gs-search-empty-->
  <div role="toolbar" class="btn-toolbar gs-search-toolbar">
    <button class="btn gs-search-toolbar-previous">Newer</button>
    <button class="btn gs-search-toolbar-next">Older</button>
  </div><!--gs-search-toolbar-->
</div><!--gs-search-->
```

During the *creation* of the search widget jQuery is used to add some functionality to the items.

1.2.1 Search Results

The JavaScript calls the AJAX Page. The results returned by the page will be displayed in the `.gs-search-results` element. To be processed properly the results have to conform to the following HTML:

- Result: `.gs-search-result`
  - Keywords [1]: `.gs-search-keyword`

The result may also be marked with the optional `.gs-search-sticky` class [2].

1.3 AJAX Page

The AJAX page is provided by products **other** than this one. When the user interacts with the HTML the JavaScript makes a `POST` query passing the following values:

`i`: The *index* (or *offset*) into the search-results.

`l`: The number of results to return (the *length*).

`s`: The text to *search* for [1].

> **Note** The AJAX pages **must** conform to this API. Other arguments to the AJAX page can be passed in as the `additionalQuery` argument during the *creation* of the search-widget.

The HTML returned by the page *must* contain *Search Results* that conform to the standard markup.

1.4 Changelog

1.4.1 2.3.2 (2016-02-01)

- Using Sphinx for the documentation
- Moving the documentation to Read the Docs
- Cleaning up the JavaScript, so it passes the Google Closure linter

1.4.2 2.3.1 (2014-10-10)

- Pointing at GitHub
- Naming the reStructuredText files as such

1.4.3 2.3.0 (2014-03-13)

- Fixing an issue with Microsoft Internet Explorer 8
- Switching to `"use strict";`

[1] The keywords are optional.
[2] The sticky results are shown first. They need to be known for the calculation for the *Next* button.
[1] If the *AJAX page* does not support searching then the HTML should be modified so the search-button is within a `div` element with the `display:none;` style set.

1.4.4 2.2.0 (2013-11-26)

- Handling no results better
- Updating the documentation

1.4.5 2.1.1 (2013-05-30)

- Following jQuery to its new home (`gs.content.js.jquery.base`)

1.4.6 2.1.0 (2013-02-26)

- Disabling the *Prev* and *Next* buttons by default
- Adding some WAI-ARIA support

1.4.7 2.0.1 (2012-12-17)

- Fixing an error with Microsoft Internet Explorer 7

1.4.8 2.0.0 (2012-06-06)

- Initial version. Refactored from similar code in `gs.group.messages.topic`, `gs.group.messages.posts`, and `gs.group.messages.files`

GroupServer uses a standard widget to provide search. By standardising the widget the interface is more consistent for the users, the pages are faster because fewer requests have to be made for the JavaScript code, and coding errors are reduced. The search code is split in to three main components.

1. This product supplies a JavaScript class that is used to provide some standard behaviour for the different search interfaces.
2. Each subsystem that provides search results produces a skeleton of HTML that works with this module to display the standard search-widget.
3. The same subsystems produce an AJAX Page that conforms to the API outlined here.

1.5 Indices and tables

- genindex
- modindex
- search

1.6 Resources

- Code repository: https://github.com/groupserver/gs.search.base/
- Questions and comments to http://groupserver.org/groups/development/
- Report bugs at https://redmine.iopen.net/projects/groupserver/

G

L

R