API calls in snippets

Last Modified on 01/31/2024 11:53 am EST

Sometimes you might want to extend KnowledgeOwl's built in functionality by utilizing client side API calls in your knowledge base.

Previously, this involved constructing an AJAX call that contains a KnowledgeOwl API key for authentication. However, by exposing your API key on the client side (even when restricted to GET access only), you are opening yourself up to unintended reader behavior.

KnowledgeOwl API Merge Code

To prevent this issue, we have added the ability to construct an API merge code within a snippet's content. On page render, the merge code will be replaced with a unique, single use URL that does not contain your API key or any account specific information.

Constructing the Merge Code

Let's take a look at how to construct this merge code and then what we can do with it. Below is the template for the merge code.

```
[ko_api(API Object|{JSON API filter})]
```

If we break the above template down into its parts, we get the following 3 required pieces.

1. **The outer wrapper: [ko_api( )]**
   This wrapper and everything within it will be replaced server side with a unique URL at the time of page rendering.

2. **The API Object**
   The first part of the inner required information denotes which API object you are going to be querying. For example, if you want query for categories, you would use **category** followed by the pipe symbol **|** .

3. **JSON API Filter**
   The second part of the inner required information needs to be a JSON formatted string containing a valid API filter. Let's say we want to query for the 5 newest categories in our knowledge base that aren't deleted. We can construct our JSON string like so:

   ```
   //project_id = Knowledge base ID
   {"project_id": "123456", "status": "active", "limit": 5, "sort": {"date_created": 1}}
   ```

When we put the parts from above together, we get the following fully constructed merge code:

```
[ko_api(category|{"project_id": "123456", "status": "active", "limit": 5, "sort": {"date_created": 1}})]
```

Knowledge Base Variables

The API merge code is replaced server side so you will not be able to use Javascript variables within it. However, we have created the following variables that you can use to reference information about the current page and the current reader that is logged in:

Variable Name	Variable Value	Example JSON
%cur_kb_id%	The ID of the knowledge base that is currently being viewed.	"project_id": "%cur_kb_id%"
%cur_cat_id%	If viewing a category: returns the ID of the category currently being viewed; If viewing an article: returns the ID of the category in which the article is contained. If the article is in a subcategory, this is the category *immediately above* this article in the hierarchy, not it's ultimate top-level parent.	"category": "%cur_cat_id%"
%cur_top_cat_id%	The ID of the top most parent category that the current article or category is in.	"category": "%cur_top_cat_id%"
%cur_parent_cat_ids%	Array of all parent category IDs that the current article or category is in.	"category": {"$in": "%cur_parent_cat_ids%"}
%cur_art_id%	The ID of the article that is currently being viewed	"id": "%cur_art_id%"
%cur_art_tags%	Array of tag IDs that are in use on the currently viewed article	"tags": {"$in": "%cur_art_tags%"}
%cur_art_permalink%	The permalink of the currently viewed article	"url_hash": "%cur_art_permalink%"
%cur_reader_id%	The ID of the currently logged in reader; will filter results by content that reader has access to; will not work for authors who are also readers	"reader_id": "%cur_reader_id%"

Variable Name	Variable Value	Example JSON
%cur_reader_groups%	Array of reader groups IDs that the currently logged in reader is assigned to	"reader_roles": "%cur_reader_groups%" *NB: The merge code itself surrounded by " is all you need - an $in comparison is already included for you in the merge code*
%cur_reader_username%	The username of the currently logged in reader	"username": "%cur_reader_username%"
%cur_search_term%	The "phrase" parameter in the URL	"phrase": "%cur_search_term%"

Using the Merge Code

Now that we have our merge code, let's look at how we can use it within our snippet content to get the information requested. Below is a script that console logs the information returned from our API call.

```
<script>
 $(function() {
  $.get('[ko_api(category|{"project_id": "123456", "status": "active", "limit": 5, "sort": {"date_created": 1}})]',
  function(apiData) {
       //do something with the returned data
       console.log(apiData);
  }).fail(function(error) {
    //uh oh something went wrong. Alert the end-user or otherwise handle the error
  });
 });
</script>
```

As you can see, the merge code is used in place of the AJAX URL, but the rest of the jQuery code remains exactly the same. When the above code is rendered to the page, the merge code is replaced with a safe, valid URL, and results in something like the following.

```
<script>
 $(function() {
  $.get('/help/ko-api/mid/9999aaaaadsfsdfsdf',
  function(apiData) {
       //do something with the returned data
       console.log(apiData);
  }).fail(function(error) {
    //uh oh something went wrong. Alert the end-user or otherwise handle the error
  });
 });
</script>
```

Now let's use some of the knowledge base variables listed above to get all of the other articles that are in the currently viewed article's category.

```
<script>
 $(function() {
  //get all published or needs review articles in the current category except for the one currently being viewed
  $.get('[ko_api(article|{"project_id": "%cur_kb_id%", "status": {"$in": ["published", "review"]}, "category": "%cur_cat_id%",
  function(apiData) {
      //do something with the returned data
      console.log(apiData);
  }).fail(function(error) {
   //uh oh something went wrong. Alert the end-user or otherwise handle the error
  });
 });
</script>
```

Working with article status

If you're pulling a list of articles via API snippet, the odds are pretty good that you're going to be using the status field. While most of our other API endpoints have a status field that is "active" or "deleted", the publishing status on articles has two statuses that could be considered active: Published ("published" in the API) and Needs Review ("review" in the API).

If you'd like to filter your article API call to get status, instead of using "status": "active" here, you'd want to use an in operator and look for the status to be in one of those two: `"status": {"$in": ["published", "review"]}` **. You can see an example of this in action in the final code block in the section before this one.**

API calls with paged results

Sometimes your API call may have multiple pages of results. In this case, we will return the next API call URL as part of the returned data. The URL will be located in the "page_stats" array like so:

```
page_stats: {
  total_records: 203
  total_pages: 3
  next_page: 2
  next_page_url: /help/ko-api/mid/9999aaaaadsfsdfsdf
}
```

Here's a template to get you started with paged API snippet calls:

```
<script>
 $(function(){
  //first page of results API call
  var firstUrl = '[ko_api(article|{"project_id": "%cur_kb_id%","_fields": ["name"], "limit": 75})]';

  //function to get multiple pages of results from API
  var getArticles = function(curUrl) {
   $.get(curUrl, function(data) {
    console.log(data);
    $.each(data['data'], function(index, value){
     //do something with api objects
    });
    //now fetch the next page of results if there is one
    //using the URL returned from the previous API call
    if(data['page_stats']['next_page_url'])
     getArticles(data['page_stats']['next_page_url']);
   }).fail(function(error) {
//you failed!
    console.log(error);
   });
  }

  //get the first page of results;
  getArticles(firstUrl);
 });
</script>
```

Requirements for use

API merge codes can ONLY be used for GET calls. Attempts to POST, PUT, or DELETE will return an error.

You must have at least one active API key in your account with ONLY GET permission. If you do not have an available API key that meets this requirement, the merge code URL will return an error.

The JSON string containing the API filter must contain a valid knowledge base ID in the format of {"project_id": "1234"}.

DO NOT include your API key in the merge code JSON. If you include an API key in the JSON, the merge code URL will return an error.

API calls in snippets do not show in article Preview mode. You'll need to publish the article to view the results of the API snippet.