Posted in: API (/spaces/127/sandpiper/forums/5044/api)

URL Query (Sorting, Filtering, etc.)

Unsubscribe from this discussion
Subscribe to RSS (../../../../../../spaces-cf/forums/rss-space-posts.ashx?spaceID=127&topicID=6006&forumID=5044&key=rrer%2B1xDo%2BlxpC7jBRbM5w%3D%3D)

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
6/12/2020

Working on the admin (https://marmelab.com/react-admin/) piece, it's clear we need to re-design some of the API, especially for **returning lists** of a resource. We currently have no support for filters or (user defined) sorts, for example.

Method name	API call
`getList`	`GET http://my.api.url/posts?sort= (http://my.api.url/posts?sort=)["title","ASC"]`
`getOne`	`GET http://my.api.url/posts/123 (http://my.api.url/posts/123)`
`getMany`	`GET http://my.api.url/posts?filter={ (http://my.api.url/posts?filter={)"id":[123`
`getManyReference`	`GET http://my.api.url/posts?filter={ (http://my.api.url/posts?filter={)"author_i`
`create`	`POST http://my.api.url/posts/123 (http://my.api.url/posts/123)`
`update`	`PUT http://my.api.url/posts/123 (http://my.api.url/posts/123)`
`updateMany`	Multiple calls to `PUT http://my.api.url/posts/123 (http://my.api.url/posts/123)`
`delete`	`DELETE http://my.api.url/posts/123 (http://my.api.url/posts/123)`
`deteleMany`	Multiple calls to `DELETE http://my.api.url/posts/123 (http://my.api.url/posts/123`

The "Method name" are the calls we need to support. The "API call" is an example implementation, but shows the kind of parameters it expect to be able to use.

"getManyReference" is particularly interesting and will require some refactoring. We currently include some "embedded" resources like this, but don't have a way to limit them.

api (/spaces/127/sandpiper/searchresults?keyword=%22api%22&searchtags=1)
filter (/spaces/127/sandpiper/searchresults?keyword=%22filter%22&searchtags=1)
sort (/spaces/127/sandpiper/searchresults?keyword=%22sort%22&searchtags=1)
url (/spaces/127/sandpiper/searchresults?keyword=%22url%22&searchtags=1)

1 Like

Reply (/forums/post?tid=6006&ReplyPostID=6007&SpaceID=127)

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
6/15/2020

We've added support for the following uri query parameters:

Sorting*

```
?sort=state:asc,zipcode:desc,addr
```

A missing direction defaults to ascending.

Filtering*

```
?filter=lastname:Johnson,age:39
```

All conditions within a filter (and across filters) are ANDed together. We only support equivalence (=) for now.

Pagination

```
?page=2&pagesize=30
```

Include Referenced Resource (aka "embed" or "expand")

```
/companies?include=users
```

We will document which resources can be included with each api endpoint (explicit support must be handled in code). For example:

```
q := db.Model(&companies)
if p.WantRelated("users") {
 q.Relation("Users")
}
p.AddFilter(q)
p.AddSort(q, "name")
q.Limit(p.Paging.PageSize).Offset(p.Paging.Offset())
p.Paging.Count, err = q.SelectAndCount()
```

*These can have multiple occurrences, as in ?**filter**=lastname:Johnson&**filter**=age:39

uri-params (/spaces/127/sandpiper/searchresults?keyword=%22uri-params%22&searchtags=1)

Like

Reply (/forums/post?tid=6006&ReplyPostID=6009&SpaceID=127) Answer

Luke Smith (https://autocare.communifire.com/people/autopartsource)
2/4/2021

As I've been building my own experimental client and server implementations, I've observed a need to "filter" (so-to-speak) the grain data itself when asking the server for a large grain-set. Different interactions call for escalating levels of grain detail. A set-comparison operation only cares about the list of UUIDs representing grains while other interactions need grain keys and potentially the actual payloads. There is about an 8x size reduction in the server's response when the grain ID's alone are conveyed. We originally had talked about a payload=yes|no type of query parameter for /grains and /slices. I think Doug wrote it that way into the GO reference implementation.
I propose a parameter called "detail" that would control the server's response behavior and there would be three (for now) valid values: GRAIN_ID_ONLY, GRAIN_WITHOUT_PAYLOAD and GRAIN_WITH_PAYLOAD

Here is how the most compact (ID only) scenario would look:

/base_url/v1/slices/00000000-0000-0000-0000-000000000000?detail=GRAIN_ID_ONLY

The server would produce an output like this:

```
{
 "grains": [
  "0d650bda-e268-f123-0f0f-12a599168672",
  "e19438f0-2959-47ad-a201-35c2ee81a634",
  "db0cd1ef-fbc3-4021-948b-97dbfe4e881c",
  "13a6d395-dc63-4810-a91e-9051b249ce8d",
  "cf5954d1-8e92-437b-832d-dcb3894ac04b",
  "2fdac7d8-2539-4041-8dc5-19a0d117d0aa",
  "621bda5d-4628-45ae-aa7f-2a49cb7a9a35"
 ]
}
```

Stepping up to the mid-level of detail would look like this

/base_url/v1/slices/00000000-0000-0000-0000-000000000000?detail=GRAIN_WITHOUT_PAYLOAD
would produce an output like this:

```
{
 "grains": [
  {
   "id": "0d650bda-e268-f123-0f0f-12a599168672",
   "description": "PIES - MicroGard Cabin filters",
   "slice_id": "99b2c2dc-113a-7c55-e59b-ae98ee93c052",
   "grain_key": "level-1",
   "source": "PIES_7_1_2020-11-29",
   "encoding": "raw",
   "payload": "",
   "payload_len": 11
  },
  {
   "id": "e19438f0-2959-47ad-a201-35c2ee81a634",
   "description": "PIES - MicroGard Cabin filters",
   "slice_id": "99b2c2dc-113a-7c55-e59b-ae98ee93c052",
   "grain_key": "level-1",
   "source": "Luke Smith (7-16-2018).jpg",
   "encoding": "z64",
   "payload": "",
   "payload_len": 2069596
 }, .....
```

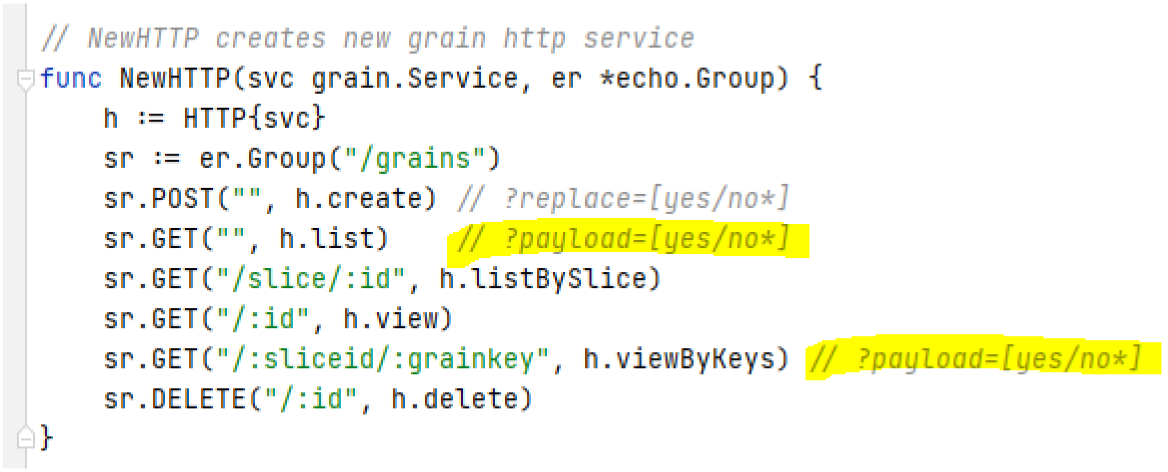
The highest level of detail would include the payload (instead of an empty string) in the *payload* member.

Like

Reply (/forums/post?tid=6006&ReplyPostID=6360&SpaceID=127)   Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
20 days ago

Yes that's correct, the Go implementation currently supports the ability to include or exclude payloads from the results:



```
// NewHTTP creates new grain http service
func NewHTTP(svc grain.Service, er *echo.Group) {
	h := HTTP{svc}
	sr := er.Group("/grains")
	sr.POST("", h.create) // ?replace=[yes/no*]
	sr.GET("", h.list)     // ?payload=[yes/no*]
	sr.GET("/slice/:id", h.listBySlice)
	sr.GET("/:id", h.view)
	sr.GET("/:sliceid/:grainkey", h.viewByKeys) // ?payload=[yes/no*]
	sr.DELETE("/:id", h.delete)
}
```

I like the ability to include just the grain_ids, but think it makes more sense to have the API as already implemented with `/grains/slice/:id` (listBySlice function) instead of off of the /slices endpoint. It would be a simple matter to add a parameter to limit to just the IDs.

1 Like

Reply (/forums/post?tid=6006&ReplyPostID=6456&SpaceID=127)   Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
14 days ago

The reference implementation actually uses a separate end point for the sync process (/sync):

```
sr.POST("/sync/:compid", h.start) // for secondary servers only
sr.GET("/sync", h.process) // for primary servers only
sr.GET("/sync/subs", h.subs) // get my subscriptions
sr.GET("/sync/slice/:id", h.grains) // ?brief=yes|no
```

The last one returns just the slice's grain ids (if brief=yes).

Like

Reply (/forums/post?tid=6006&ReplyPostID=6457&SpaceID=127)   Answer

Copyright © 2021 Axero Solutions LLC.
Auto Care Association powered by Communifire ™ Version 8.0.7789.8960

© 2021 - AUTO CARE ASSOCIATION (http://autocare.org) | LEGAL & PRIVACY STATEMENT (https://www.autocare.org/privacy-statement/)