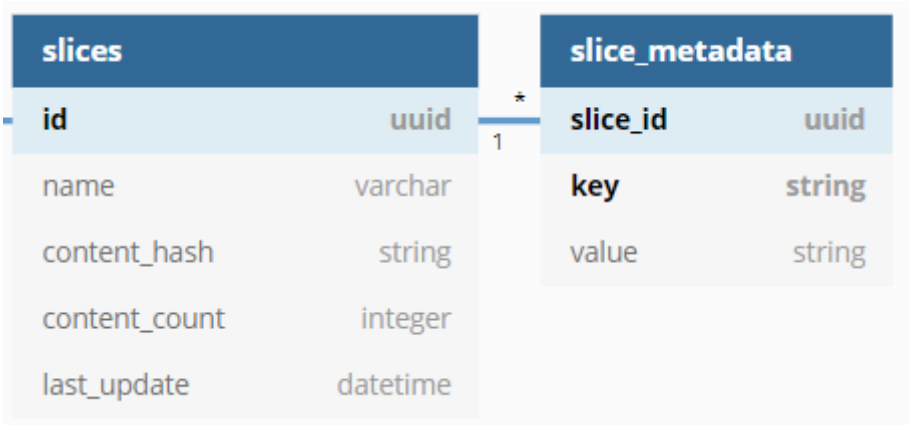
Posted in: Development (/spaces/127/sandpiper/forums/5366/development)

Slice Metadata JSON (API request and response)

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
1/4/2020

This post discusses a design choice that affects the API. I'm working on the `Slice` resource. Specifically Create (POST) and View (GET), but the pattern we decide to use here will be used everywhere.

Here is the ER diagram for the database.



As far as the API is concerned, these two tables are **handled together**. When you ask for a slice, you also get its `slice_metadata`. When you POST a new slice, you also post its metadata (which could be empty to begin with). Updates will also handle these as one resource.

The decision, then, is how to design the JSON for this resource. There are two choices I'm considering:

1. An Array of Key/Value structs:


```
{
  "id": "7b15e213-96ce-45b2-8d87-a70ec2d342e2",
  "slice_name": "Slice8",
  "content_hash": "",
  "content_count": 0,
  "last_update": "0001-01-01T00:00:00Z",
  "created_at": "2020-01-04T19:17:40.914014Z",
  "updated_at": "2020-01-04T19:17:40.914014Z",
  "metadata": [
    {
      "key": "pcdb-version",
      "val": "2019-09-27"
    },
    {
      "key": "vcdb-version",
      "val": "2019-09-27"
    }
  ]
}
```


2. A "map" (dictionary) of key/value strings:


```
{
    "id": "7b15e213-96ce-45b2-8d87-a70ec2d342e2",
    "slice_name": "AAP-Brakes",
    "content_hash": "",
    "last_update": "0001-01-01T00:00:00Z",
    "created_at": "2020-01-04T19:17:40.914014Z",
    "updated_at": "2020-01-04T19:17:40.914014Z",
    "metadata": {
        "pcdb-version": "2019-09-27",
        "vcdb-version": "2019-09-27"
    },
    "count": 2919
}
```


At the moment, I have the first option working, but I like the look of the second option. You can iterate over both of them, so they should be equally easy to work with (at least in Go).

Chime in if you have an opinion. I'll probably do some research on this to see if there's any kind of standard approach. I don't want to implement something that's difficult to work with in javascript, for example.

choices (/spaces/127/sandpiper/searchresults?keyword=%22choices%22&searchtags=1)
json (/spaces/127/sandpiper/searchresults?keyword=%22json%22&searchtags=1)
serialization (/spaces/127/sandpiper/searchresults?keyword=%22serialization%22&searchtags=1)

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
1/4/2020

Another reason I like the second option is that an array implies an order. We really don't care about order for metadata values.

In **javascript** (at least), there's no problem iterating over the second option with something like the following:

```
for (var key in metadata) {
    var val = metadata[key];
    ...
}
```

And of course, you can also access values directly like this:

```
val = metadata['vcdb-date']
metadata['vcdb-date'] = '2019-09-27'
```

I'll see about changing the code to use the second option (with a `map[string]string` instead of `[]struct`).

javascript (/spaces/127/sandpiper/searchresults?keyword=%22javascript%22&searchtags=1)

Like

Reply (/forums/post?tid=5449&ReplyPostID=5451&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
1/5/2020

Changing the code to use option 2 wasn't as easy as I thought because the ORM we're using doesn't load maps from the database automatically the way it does for arrays of structs.

This was all the code we needed before:

```
var slice = &sandpiper.Slice{ID: id}

// get slice by primary key
err := db.Select(slice)
```

But for option 2, we had to perform the secondary (metadata) lookup ourselves, and manually copy the map into the slice result (using a new "ToMap" function we wrote for this purpose).

```
// insert slice metadata into result
var meta sandpiper.MetaArray
err = db.Model(&meta).Where("slice_id = ?", id).Select()
if err != nil {
 return nil, err
}
slice.Metadata = meta.ToMap()
```

Here's the metadata structure (that matches the database table) along with the ToMap() function.

```
// SliceMetadata contains information about a slice
type SliceMetadata struct {
 SliceID uuid.UUID `json:"-" pg:",pk"`
 Key     string    `json:"key" pg:",pk"`
 Value   string    `json:"val"`
}

// MetaArray is an array of slice metadata
type MetaArray []SliceMetadata

// ToMap converts array of metadata key/value structs to a map
func (a MetaArray) ToMap() MetaMap {
 mm := make(MetaMap)
 for _, meta := range a {
  mm[meta.Key] = meta.Value
 }
 return mm
}
```

But, in the end, we got what we wanted:

```
{
  "id": "eaea8308-1840-4802-ad38-72b53e31594c",
  "slice_name": "Slice2",
  "content_hash": "",
  "content_count": 0,
  "content_date": "0001-01-01T00:00:00Z",
  "created_at": "2020-01-05T03:56:36.373565Z",
  "updated_at": "2020-01-05T03:56:36.373565Z",
  "metadata": {
    "pcdb-version": "2019-09-27",
    "vcdb-version": "2019-09-27"
  }
}
```


I'm not sure any ORM would be able to handle this any better. So this is something we might consider changing back if we run into this a lot.

orm (/spaces/127/sandpiper/searchresults?keyword=%22orm%22&searchtags=1)

Like

Reply (/forums/post?tid=5449&ReplyPostID=5455&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
1/5/2020

Similar to metadata, slices also have 0 or more "companies" assigned to them (through subscriptions). A slice JSON response will include these companies in an array. (There's no question how these should be represented, however.)

Here's a working example response:

```
{
  "id": "eaea8308-1840-4802-ad38-72b53e31594c",
  "slice_name": "Slice2",
  "content_hash": "",
  "content_count": 0,
  "content_date": "0001-01-01T00:00:00Z",
  "created_at": "2020-01-05T03:56:36.373565Z",
  "updated_at": "2020-01-05T03:56:36.373565Z",
  "metadata": {
    "pcdb-version": "2019-09-27",
    "vcdb-version": "2019-09-27"
  },
  "Companies": [
    {
      "id": "9ad07234-2742-40eb-9ef1-800c2f1164ce",
      "name": "sandpiper owner",
      "sync_addr": "https://sandpiper.example.com",
      "active": true,
      "created_at": "2020-01-04T21:16:29.660427Z",
      "updated_at": "2020-01-04T21:16:29.660427Z"
    }
  ]
}
```

If there are no subscriptions for a slice, you'd simply have an empty array:

```
Companies: []
```

We'll use a similar pattern in other areas of the system (for example, Companies could have Users embedded in the response). This can keep the client from having to do extra API calls to get commonly related information.

Here is the final code for looking up a Slice by ID (ignore the random boldness included by communifire, I have no control over this, even in the "Code Editor").

```
// View returns a single slice by ID with metadata and subscribed companies
// (assumes allowed to do this)
func (s *Slice) View(db orm.DB, id uuid.UUID) (*sandpiper.Slice, error) {
        var slice = &sandpiper.Slice{ID: id}

        // get slice by primary key with subscribed companies
        err := db.Model(slice).Relation("Companies").WherePK().Select()
        if err != nil {
                return nil, err
        }

        // insert any slice metadata into response
        var meta sandpiper.MetaArray
        err = db.Model(&meta).Where("slice_id = ?", id).Select()
        if err != nil {
                return nil, err
        }
        slice.Metadata = meta.ToMap()

        return slice, nil
}
```

api-design (/spaces/127/sandpiper/searchresults?keyword=%22api-design%22&searchtags=1)
related-data (/spaces/127/sandpiper/searchresults?keyword=%22related-data%22&searchtags=1)

Like

Reply (/forums/post?tid=5449&ReplyPostID=5456&SpaceID=127) Answer