201806-搜索客户端设计

Created by 杨超, last modified on 2018 Jun 06

目的

设计一个可以方便使用的类似ActiveQuery的客户端，可以让业务方比较简单的调用搜索的各种可用功能【包括但不仅限于 query/filter/sort/limit/offset/page等功能】

调研

1 opensearch的其他封装

主要看了laravel里的opensearch封装 https://github.com/lingxi/ali-opensearch-sdk

主要的思路

1.1 model+searchable – 每个table自定义model 并且 searchable – index/fields(或许应该加上其他配置，比如字段类型)

```php
namespace App\Models;

use Lingxi\AliOpenSearch\Searchable;

class User extends Model
{
    use Searchable;

    /**
     * Get the index name for the model.
     *
     * @return string
     */
    public function searchableAs()
    {
        return 'user';
    }

    public function toSearchableDocCallbacks($actions = ['update', 'delete'])
    {
        throw new Exception('这个应用不需要手动维护数据');
    }

    public function getSearchableFields()
    {
        return 'id';
    }
```

1.2 使用类似ActiveQuery – 这点和我想的基本一直 query/select/filter/orderby 不同点是我这边支持limit/offset/page

简单搜索

```php
<?php

$result = User::search(['name' => 'lingxi'])
    ->select([
        'id',
        'name',
        'age',
    ])
    ->filter(['age', '<', '30'])
    ->filter(['age', '>', '18'])
    ->orderBy('id', 'desc')
    ->paginate(15);
```

2 elasticsearch的封装

主要参照了yii2的elasticsearch，看起来elasticsearch因为开源和时间长而且和db结合更加紧密，功能更加强大，可以考虑下个Q尝试一下

https://github.com/yiisoft/yii2-elasticsearch/blob/master/docs/guide/mapping-indexing.md

主要还是elasticsearch本身的优势，可参考点并不是太多，唯一可用的就是解决了我一直在想如何解决字段类型定义的问题，参考下图的mapping，就是让用户自定义好

```
class Book extends yii\elasticsearch\ActiveRecord
{
    // Other class attributes and methods go here
    // ...

    /**
     * @return array This model's mapping
     */
    public static function mapping()
    {
        return [
            static::type() => [
                'properties' => [
                    'name'           => ['type' => 'string'],
                    'author_name'    => ['type' => 'string'],
                    'publisher_name' => ['type' => 'string'],
                    'created_at'     => ['type' => 'long'],
                    'updated_at'     => ['type' => 'long'],
                    'status'         => ['type' => 'long'],
                ]
            ],
        ];
    }
```

设计思路

综合调研的参考点 和 之前对opensearch的了解和思考总结

我构想中的 – 可以考虑把SearchModel放到common里面

SeachModel

1 index --> 定义表名

2 mapping --> 定义表字段的类型 int / string / int_array / string_array

3. search / suggest --> instance 默认就叫search/suggest

SearchQuery

$arr = SearchModel::search()->filter()->andFilter()->orderBy()->limit()->offset()->all();
SearchModel::search() / suggest()

->select([]) // fields字段
->query() // query词，如果为空 就需要配置默认参数 type=1 or type=2 or type=3 || 由search的服务端配置好--后续可以约定好交给客户端
->filter(['type' => new SearchInt(1)]) // filter相当于where
//'type'=>'1' 'type'=>1 bigint '1234'
->andFilter(['status' => 10]) // andWhere 组装的时候 条件合成 (type=1) and (status=10)
->orFilter(['like', 'catgy', 'bc']) // orWhere 组装的时候 条件合成 (type=1) and (status=10) or (catgy like '%bc%')
// andFilter 组装的时候 条件合成 and tag in ('8','9','10')
// 有个问题 所有的条件int和string是有区别的 需要显示的配置在searchModel里面
->andWhere(['in', 'tag', [8,9,10]])
->all();

filter比较特殊的原因

int型的是通用的
string类型的
text/shortText/literal 的名字需要不同
text 用于真正的搜索分词
shortText实现Like
literal实现 = 也就是精准匹配
array型
literal_array 实现 in/not in

Like 李伟 likes this

No labels