ElasticSearch engine adapter

This extension is an ActiveRecord-like wrapper for ElasticSearch full text search engine integration into  
the Yii2 framework. It allows you to work with any model data and use the ActiveRecord pattern to  
retrieve and store records in ElasticSearch collections.

Подготовка

1. Создайте новое приложение с помощью диспетчера пакетов Composer, как описано в официальном руководстве по адресу  
   <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.   
   По русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-installation>.
2. Install the ElasticSearch service found at <https://www.elastic.co/downloads/elasticsearch>.
3. Install the extension with the following command:

Composer require yiisoft/yii2-elasticsearch

How to do it...

Set the new ElasticSearch connection in your application configuration:

return [

//....

'components' => [

'elasticsearch' => [

'class' => 'yii\elasticsearch\Connection',

'nodes' => [

['http\_address' => '127.0.0.1:9200'],

// configure more hosts if you have a cluster

],

],

]

];

Using the Query class

You can use the Query class for the low-level querying of records from any collection:

use \yii\elasticsearch\Query;

$query = new Query;

$query->fields('id, name')

->from('myindex', 'users')

->limit(10);

$query->search();

You can also create a command and run it directly:

$command = $query->create  
Command();

$rows = $command->search();

Using ActiveRecord

Using ActiveRecord is a common way to access your records. Just extend the

yii\elasticsearch\ActiveRecord class and implement the attributes() method to define the attributes  
of your documents.

For example, you can write the Customer model:

class Buyer extends \yii\elasticsearch\ActiveRecord  
{

public function attributes()

{

return ['id', 'name', 'address', 'registration\_date'];

}

public function getOrders()

{

return $this->hasMany(Order::className(), ['buyer\_id' => 'id'])->orderBy('id');

}

}

Then write the Order model:

class Order extends \yii\elasticsearch\ActiveRecord  
{

public function attributes()

{

return ['id', 'user\_id', 'date'];

}

public function getBuyer()

{

return $this->hasOne(Customer::className(), ['id' => 'buyer\_id']);

}

}

You may override index() and type () to define the index and type this record represents.

The following is a usage example:

$buyer = new Buyer();

$buyer>primaryKey = 1; // it equivalent to $customer->id = 1;

$buyer>name = 'test';

$buyer>save();

$buyer = Buyer::get(1);

$buyer = Buyer::mget([1,2,3]);

$buyer = Buyer::find()->where(['name' => 'test'])->one();

You can use Query DSL for specific queries:

$result = Article::find()->query(["match" => ["title" => "yii"]])->all();

$query = Article::find()->query([

"fuzzy\_like\_this" => [

"fields" => ["title", "description"],

"like\_text" => "Some search text",

"max\_query\_terms" => 12

]

]);

$query->all();

You can add facets to your search:

$query->addStatisticalFacet('click\_stats', ['field' => 'visit\_count']);

$query->search();

Using the ElasticSearch DebugPanel

This extension contains a special panel for the yii2 - debug module. It allows you to view all executed  
queries. You can include this panel in your configuration file:

if (YII\_ENV\_DEV) {

// configuration adjustments for 'dev' environment  
$config['bootstrap'][] = 'debug';

$config['modules']['debug'] = [

'class' => 'yii\debug\Module',

'panels' => [

'elasticsearch' => [

'class' => 'yii\elasticsearch\DebugPanel',

],

],

];

$config['bootstrap'][] = 'gii';

$config['modules']['gii'] = 'yii\gii\Module';

}

How it works...

The extension provides a low-level command builder and high-level ActiveRecord implementation for  
querying records from the ElasticSearch index.

The extension’s ActiveRecord usage is very similar to the database ActiveRecord as described in Chapter  
3, ActiveRecord, Model, and Database, besides the join(), groupBy(), having(), and union()  
ActiveQuery operators.

Note

Note: ElasticSearch limits the number of returned records to ten items by default. Take care with limits  
if you use relations with the via() option.

See also

* For more information about the extension, see:

° [https://github.com/yiisoft/yii2-elasticsearch/hlob/master/docs/guide/RFADMF.md](https://github.com/yiisoft/yii2-elasticsearch/blob/master/docs/guide/README.md)  
o [http://www.viiframework.com/doc-2.0/ext-elasticsearch-index.html](http://www.yiiframework.com/doc-2.0/ext-elasticsearch-index.html)

* You can also visit the official extension site at <https://www.elastic.co/products/elasticsearch>.
* For more information about Query DSL, you can visit:

° <http://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query.html>  
o [http://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-flt-quervhtml](http://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-flt-query.html)

* For ActiveRecord usage refer to the Chapter 3, ActiveRecord, Model, and Database