MongoDB driver

This extension provides the MongoDB integration for the Yii2 framework and allows you to work with  
MongoDB collection’s records via the ActiveRecord-style model.

Getting ready

1. Create a new application by using composer, as described in the official guide at  
   [http://www.yiiframework.eom/doc-2.0/guide-start-installation.html](http://www.yiiframework.com/doc-2.0/guide-start-installation.html).
2. Install MongoDB using the correct installation process from  
   [https ://docs.mongodb. org/ma nual/installation/](https://docs.mongodb.org/manual/installation/) for your system.
3. Install the php5-mongo PHP extension.
4. Install the component with the following command:

composer require yiisoft/yii2-mongodb

How to do it...

1. First of all, create the new MongoDB database. Run it in the mongo-client shell and type the  
   database name:

mongo

> use mydatabase

1. Add this connection information to your components config section:

return [

// ...

'components' => [

// ...

'mongodb' => [

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

'dsn' =>

'mongodb://localhost:27017/mydatabase',

],

],

];

1. Add the new console controller to your console configuration file:

return [

// ...

'controllerMap' => [

'mongodb-migrate' =>

'yii\mongodb\console\controllers\MigrateController'

],

];

1. Create the new migration with the shell command:

php yii mongodb-migrate/create create\_customer\_collection

1. Type the following code into the up() and down () methods:

<?php

use yii\mongodb\Migration;

class m160201\_102003\_create\_customer\_collection extends Migration  
{

public function up()

{

$this->createCollection('customer');

}

public function down()

{

$this->dropCollection('customer');

}

}

1. Apply the migration:

php yii mongodb-migrate/up

1. Put the MongoDB debug panel and models generator into your configuration:

if (YII\_ENV\_DEV) {

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

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

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

'panels' => [

'mongodb' => [

'class' => 'yii\mongodb\debug\MongoDbPanel',

],

],

];

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

$config['modules']['gii'] = [

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

'generators' => [

'mongoDbModel' => [

'class' => 'yii\mongodb\gii\model\Generator'

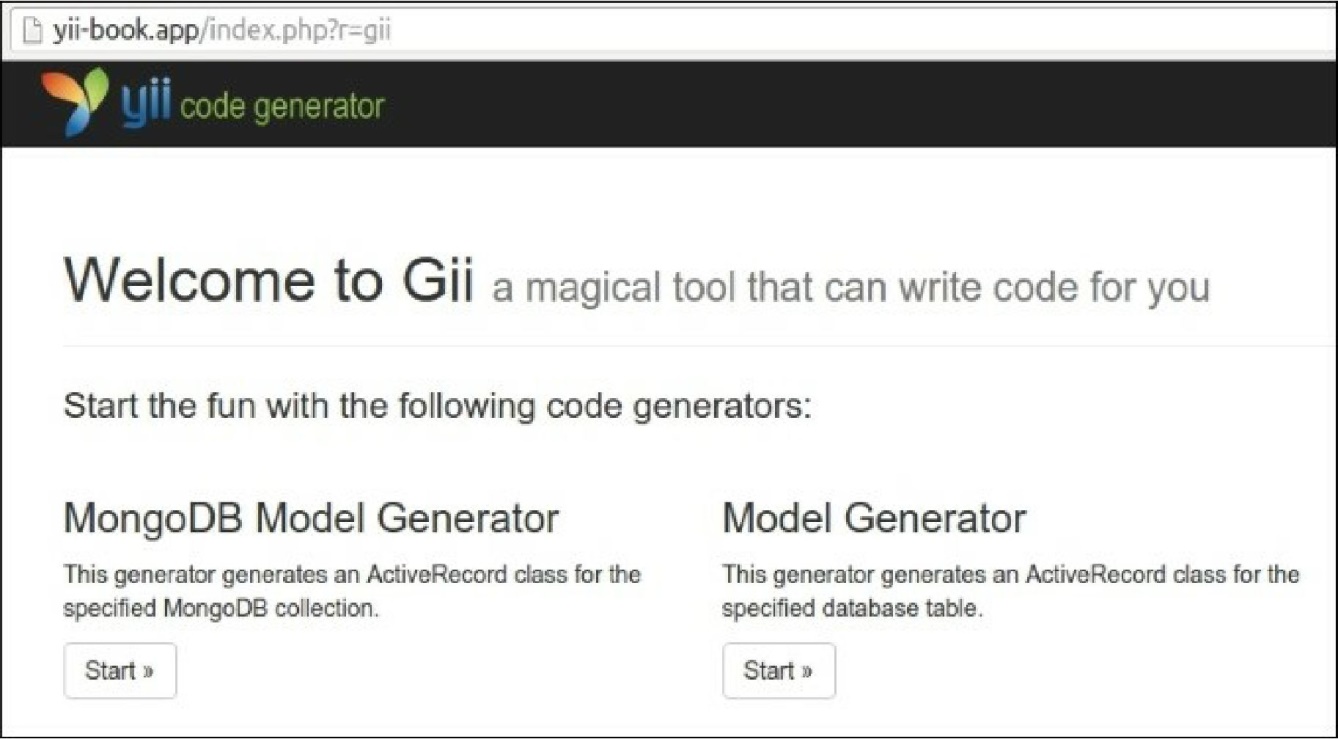
]

],

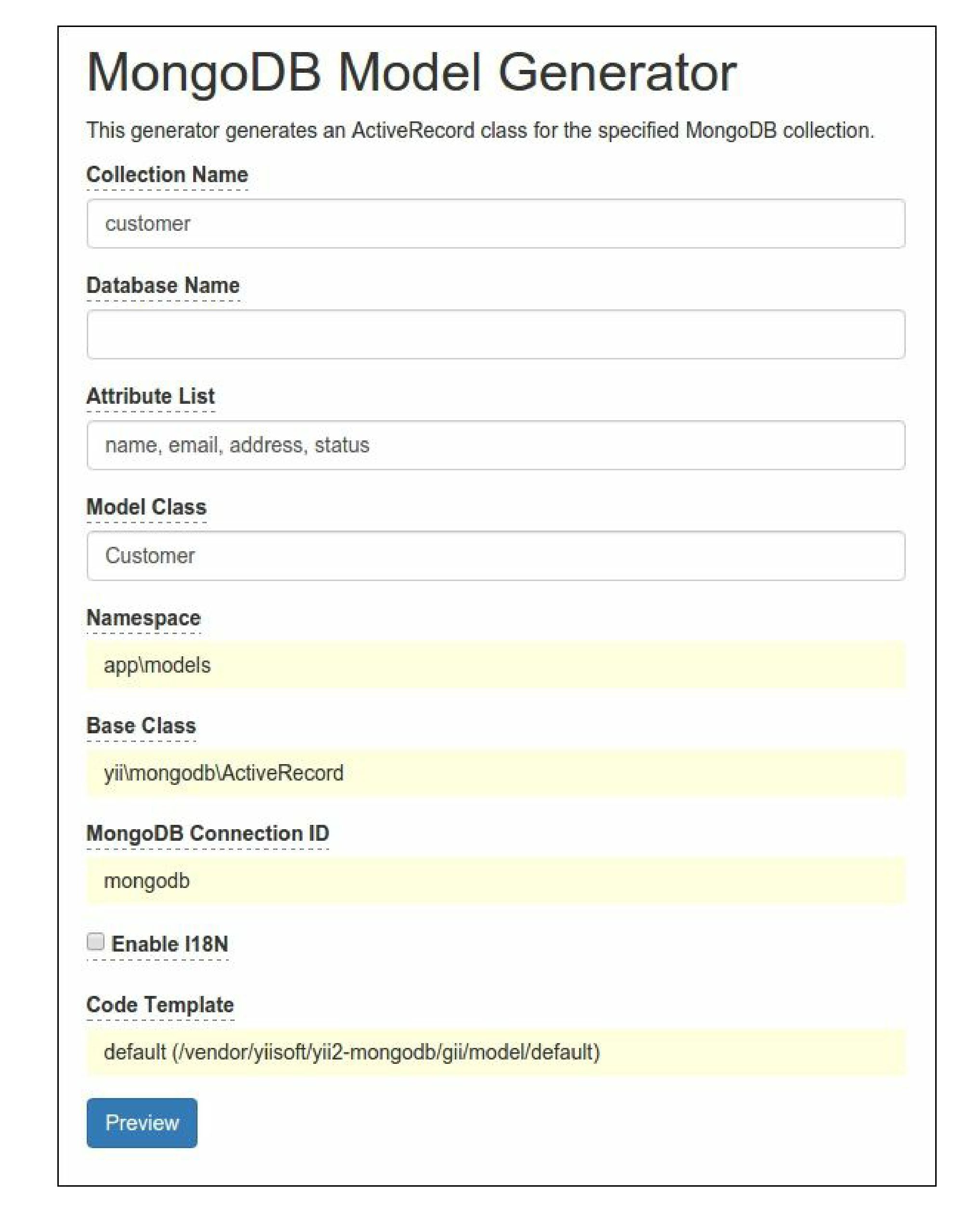
];

}

1. Run the Gii generator:



9. Start the new MongoDB Model Generator to generate the new model for your own collection:



1. Click the Preview and Generate buttons.
2. Check that you have the new model, app\models\customer:

<?php

namespace app\models;  
use Yii;

use yii\mongodb\ActiveRecord;

/\*\*

* This is the model class for collection "customer".

\*

* @property \MongoId|string $\_id
* @property mixed $name
* @property mixed $email
* @property mixed $address
* @property mixed $status  
  \*/

class Customer extends ActiveRecord  
{

public static function collectionName()

{

return ' customer ' ;

}

public function attributes()

{

return [

'\_id',

'name',

'email',

'address',

'status',

];

}

public function rules()

{

return [

[['name', 'email', 'address', 'status'], 'safe']  
];

}

public function attributeLabels()

{

return [

'\_id' => 'ID',

'name' => ' Name',

'email' => 'Email',

'address' => 'Address',

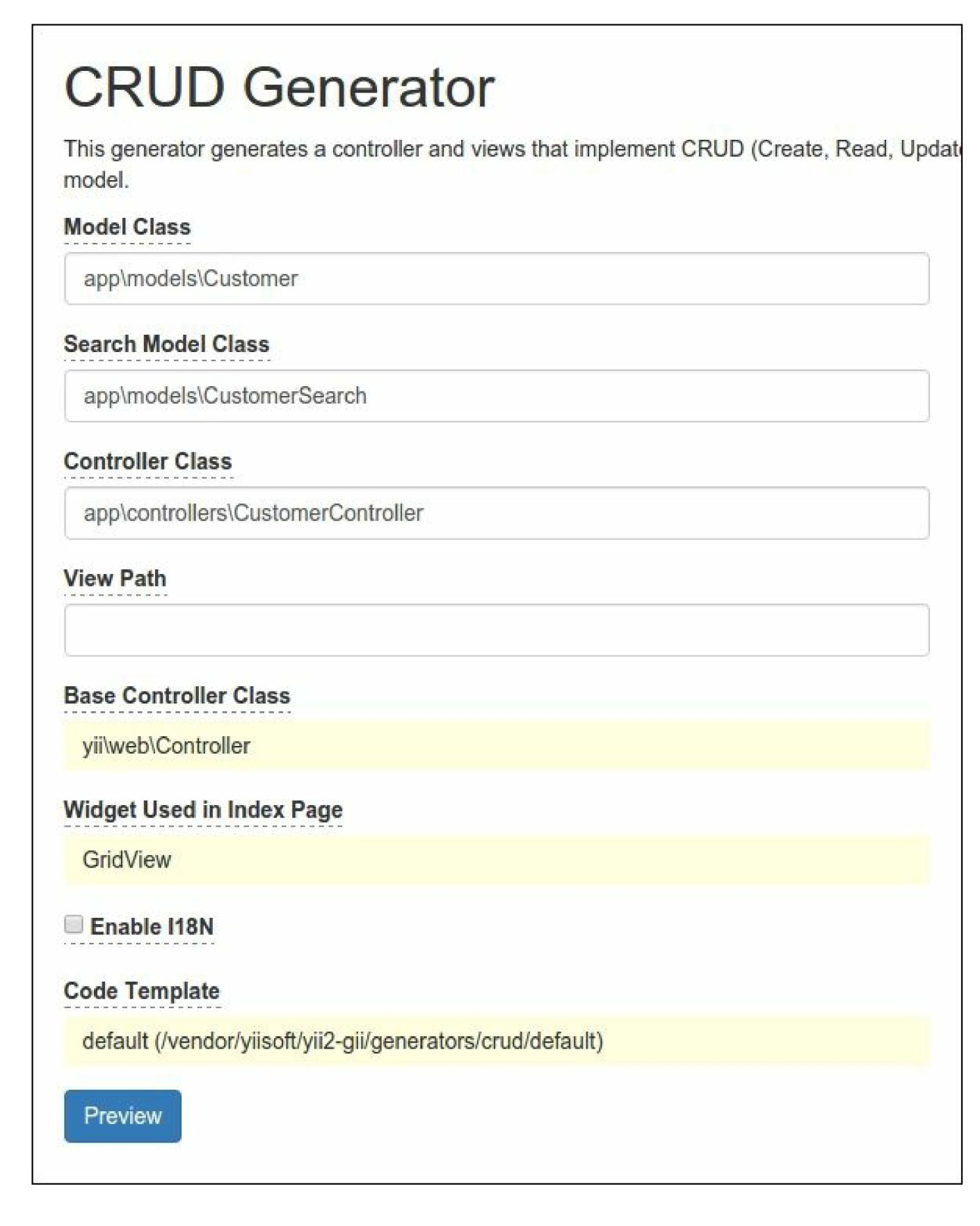
'status' => 'Status',

];

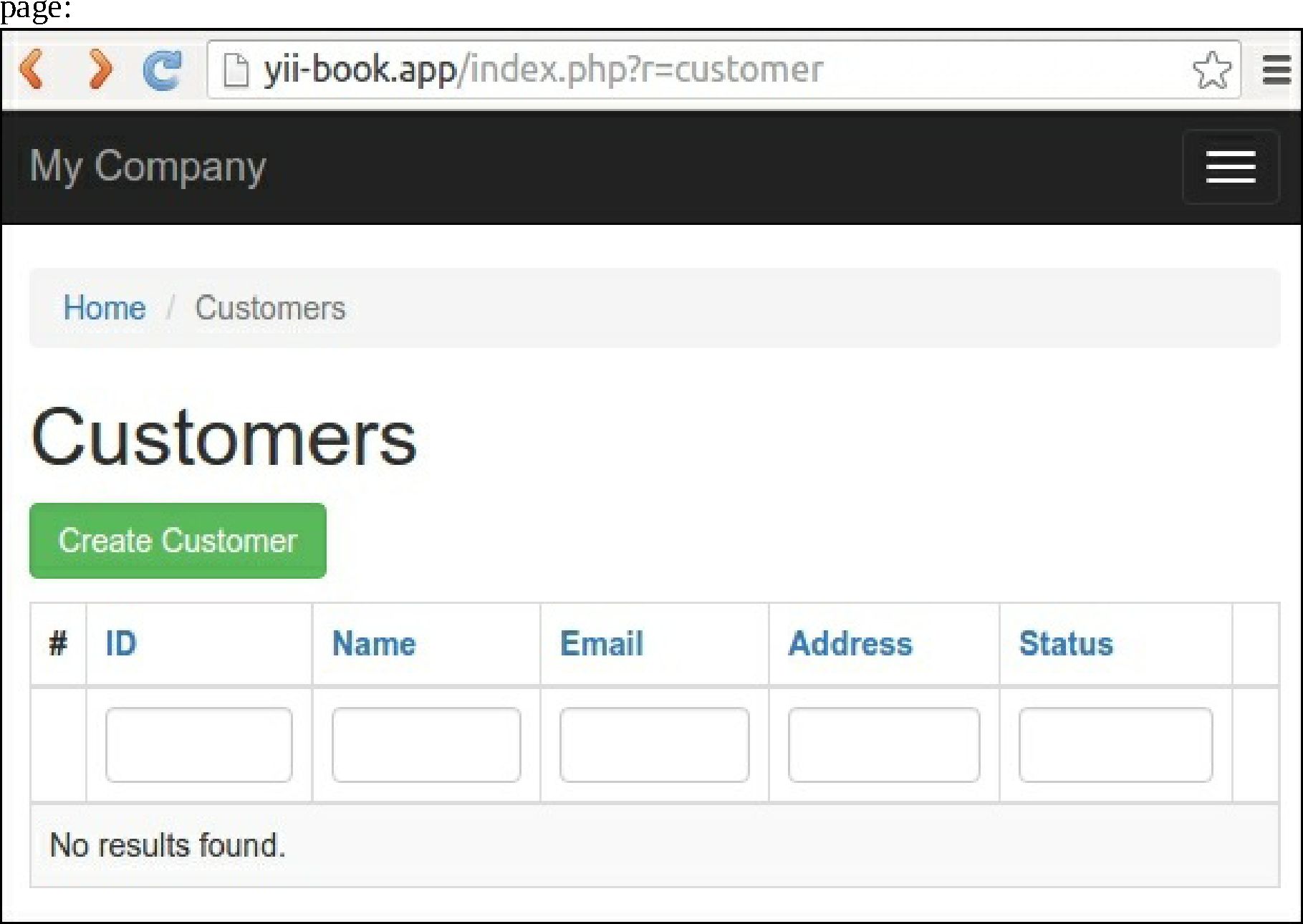
}

}

1. Run Gii again and generate the CRUD:



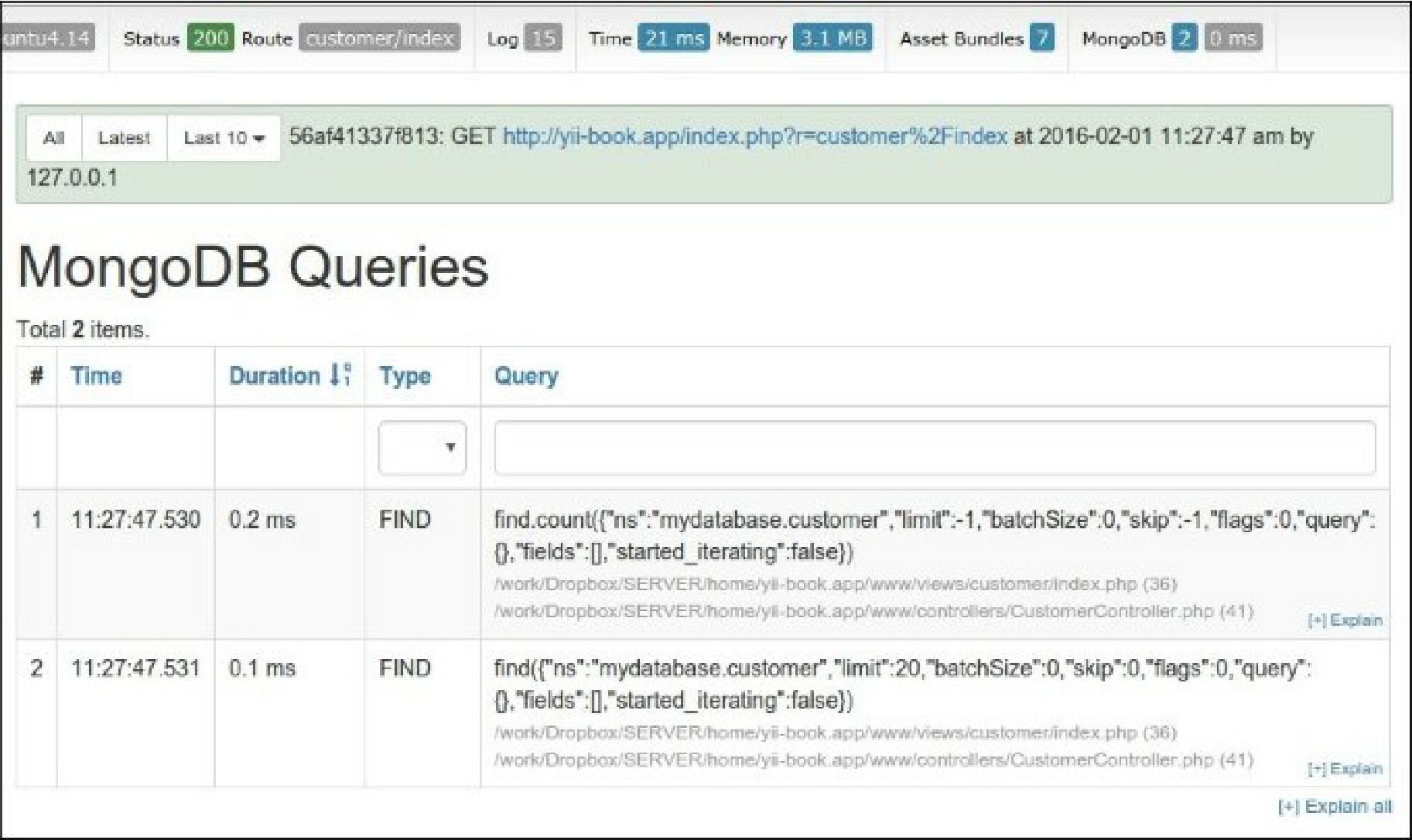
13. Check that you have generated the Customercontroller class and run the new customer manager



1. You may create, update, and delete your customers’ data right now.
2. Look for the Debug panel in the page footer:



16. You can see the total MongoDB query count and total execution time. Click on the count badge and  
inspect the queries:



Basic usage

You may access databases and collections via the \yii\mongodb\collection instance:

$collection = Yii::$app->mongodb->getCollection('customer');$collection->insert(['name'  
=> 'John Smith', 'status' => 1]);

To perform the find queries, you should use \yii\mongodb\Query:

use yii\mongodb\Query;

$query = new Query;

// compose the query  
$query->select(['name', 'status'])

->from('customer')

->limit(10);

// execute the query  
$rows = $query->all();

Note

Note: The MongoDB document id ("\_id" field) is not scalar, but an instance of the \MongoId class.

You must not care about the conversion from integer or string $id values to \MongoId, because query  
builder converts it automatically:

$query = new \yii\mongodb\Query;

$row = $query->from('item')

->where(['\_id' => $id]) // implicit typecast to \MongoId  
->one();

To get the actual Mongo ID string, you should typecast the \Mongoid instance to a string:

$query = new Query;

$row = $query->from('customer')->one();

var\_dump($row['\_id']); // outputs: "object(MongoId)"var\_dump((string)$row['\_id']);

How it works...

The Query, ActiveQuery, and ActiveRecord classes of this extension extends yii\db\QueryInterface  
and yii\db\BaseActiveRecord, and therefore they are compatible with the built-in framework Query,  
ActiveQuery, and ActiveRecord classes.

You can use the yii\mongodb\ActiveRecord class for your models and the yii\mongodb\ActiveQuery  
builder to retrieve your models and use them in your data provider:

use yii\data\ActiveDataProvider;  
use app\models\Customer;

$provider = new ActiveDataProvider([

'query' => Customer::find(),

'pagination' => [

'pageSize' => 10,

]

]);

For general information on how to use Yii’s ActiveRecord, please refer to the Chapter 3, ActiveRecord,  
Model, and Database.

See also

* For more information about the extension, refer to the following URLs:

° [https://github.com/yiisoft/yii2-mongodb/hloh/master/docs/gnide/RFADME.md](https://github.com/yiisoft/yii2-mongodb/blob/master/docs/guide/README.md)  
° <http://www.yiiframework.com/doc-2.0/ext-mongodb-index.html>

* And for information about the original library, refer to:

o [httpsy/docs.mongodb.org/maniial/](https://docs.mongodb.org/manual/)

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