Query Builder

* Cake\ORM\Query
* https://book.cakephp.org/3.0/en/orm/query-builder.html

The Query Object

* Using Cake\ORM\TableRegistry

$articles = TableRegistry::get(‘Articles’);

// find all

$query = $articles->find();

* Inside ArticlesController.php

$query = $this->Articles->find();

* Bình thường khi find() xong thì check sql ở debug kit thấy nó vẫn không call sql, cho nên cần foreach nó thì nó mới get result về
* Method:
* debug($query);
* Get raw sql, but don’t show params

$sql = $query->sql();

Selecting Rows From A Table

* Find

->find()

* Select

->select([‘colum-1’, ‘colum-2’])

* Select with alias column

$query->select(['pk' => 'id', 'aliased\_title' => 'title', 'body']);

// Results in SELECT id AS pk, title AS aliased\_title, body

* Select all fields

$query->select($users); // select all fields from users

* Distinct

->distinct([‘name’])

* Where
* Where =

->where([‘id’ => 1])

* Where >, <….

->where([‘id >’ => 1]

* And where

->andWhere([‘name’ => ‘npt’]);

* Or where

->orWhere([‘name’ => ‘haha’])

* Example where:
* Ex 1

$query = $articles->find()

->where(['title LIKE' => '%First%'])

->andWhere(function ($exp) {

return $exp->or\_([

'author\_id' => 2,

'is\_highlighted' => true

]);

});

// out

WHERE (

title LIKE '%First%'

AND

(author\_id = 2 OR is\_highlighted = 1)

)

* Ex2:

$query = $articles->find()

->where(function ($exp) {

return $exp

->eq('author\_id', 2)

->eq('published', true)

->notEq('spam', true)

->gt('view\_count', 10);

});

//out

SELECT \*

FROM articles

WHERE (

author\_id = 2

AND published = 1

AND spam != 1

AND view\_count > 10)

* The expression object
* eq()
* notEq()
* like()

$exp->like(‘name’, ‘%A%’)

// WHERE name LIKE "%A%"

* notLike()

$exp->notLike('name', '%A%');

// name NOT LIKE "%A%"

* in()

$exp->in('country\_id', ['AFG', 'USA', 'EST'])

// country\_id IN ('AFG', 'USA', 'EST')

* notIn()
* gt()

$exp->gt('population', '10000')

// population > 10000

* gte()

$exp->gte('population', '10000')

// population >= 10000

* lt()

$exp->lt('population', '10000');

// population < 10000

* lte()

$exp->lte('population', '10000')

// population <= 10000

* isNull()
* isNotNull
* between()

$exp->between('population', 999, 5000000);

// population BETWEEN 999 AND 5000000

* exists()
* notExists()
* Group by

->group(‘name’)

* Having

->having([‘id >’ => 3])

* Order by

->order([‘created’ => ‘DESC’])

* In 3.0.12, addition to order, orderAsc, orderDesc
* Contain: relashionship with Comments Authors

$query->contain(['Comments', 'Authors'])

* Passing conditions to contain: https://book.cakephp.org/3.0/en/orm/retrieving-data-and-resultsets.html#passing-conditions-to-contain

// In a controller or table method.

// Prior to 3.5.0 you would use contain(['Comments' => function () { ... }])

$query = $articles->find()->contain('Comments', function ($q) {

return $q

->select(['body', 'author\_id'])

->where(['Comments.approved' => true]);

})

* For pagination ad the Controller

$this->paginate['contain'] = [

'Comments' => function (\Cake\ORM\Query $query) {

return $query->select(['body', 'author\_id'])

->where(['Comments.approved' => true]);

}

];

* It is also possible to restrict deeply-nested associations using the dot notation:

$query = $articles->find()->contain([

'Comments',

**'Authors.Profiles'** => function ($q) {

return $q->where(['Profiles.is\_published' => true]);

}

]);

* You can tell contain() to not append the foreignKey constraints to the generated query. In that case you shout use an array passing foreignkey and queryBuilder

$query = $articles->find()->contain([

'Authors' => [

'foreignKey' => false,

'queryBuilder' => function ($q) {

return $q->where(...); // Full conditions for filtering

}

]

]);

* Maching -> where relationshiop

// get permissions from roleId[]

$permission = TableRegistry::get('Permissions')->find()->matching('RolePermissions', function ($q) use($roleId){

return $q->where(function($exp) use($roleId){

return $exp->in('role\_id', $roleId);

});

})->all();

* Limit

->limit(50)

* Paginate with page 2

->page(2)

* Execute sql and return result
* Get all

->all();

* Get first

->first();

* Get all change To Array

->toArray();