Query Builder - Codeigniter 3

$this->db->get()

Executa a seleção consulta e retorna o resultado. Pode ser usado por si para recuperar todos os registros de uma tabela:

```
$query = $this->db->get('minhaTabela');//SELECT * FROM minhaTabela
```

O segundo e terceiro parâmetros permitem definir um limite e cláusula de compensação:

```
$query = $this->db->get('minhaTabela', 10, 20); //SELECT * FROM minhaTabela LIMIT 20, 10
```

Você vai notar que a função acima é atribuído a uma variável chamada `$query`, que pode ser usado para mostrar os resultados:

```
$query = $this->db->get('minhaTabela');
foreach($query->result() as $linha){
    echo $linha->titulo;
}
```

$this->db->get_compiled_select()

Compila a seleção consulta apenas como `$this->db->get()` , mas não executar a consulta. Este método simplesmente retorna a consulta SQL como uma string:

```
$sql = $this->db->get_compiled_select('minhaTabela');
echo $sql; //"SELECT * FROM minhaTabela"
```

O segundo parâmetro permite definir ou não a consulta construtor de consulta será reposto (por padrão, ele será zerado, assim como quando usando `$this->db->get()`):

```
echo $this->db->limit(10, 20)->get_compiled_select('minhaTabela', FALSE); //"SELECT * FROM LIMIT
mytable 20, 10"

echo $this->db->select('titulo, conteudo, data')->get_compiled_select(); //"SELECT * FROM minhaTa
bela LIMIT 20, 10"
```

A principal coisa a notar no exemplo acima é que a segunda consulta não utilizar `$this->db->from()` e não passar um nome de tabela para o primeiro parâmetro. A razão para este resultado é porque a consulta não foi executada usando `$this->db->get()` que redefine valores ou redefinir directamente utilizando `$this->db->reset_query()` .

$this->db->get_where()

Idêntico à função acima, exceto que ele permite que você adicione uma cláusula “where” no segundo parâmetro, em vez de usar o db->where():

```
$query = $this->db->get_where('minhaTabela', array('id' => $id), $limit, $offset);
```

$this->db->select()

Permite-lhe para escrever a parte SELECT da sua consulta:

```
$this->db->select('titulo, conteudo, data');
$query = $this->db->get('minhaTabela');
//SELECT titulo, conteudo, data FROM minhaTabela
```

`$this->db->select()` aceita um segundo parâmetro opcional. Se você configurá-lo para FALSE, CodeIgniter não vai tentar proteger seu campo ou tabela nomes. Isso é útil se você precisar de uma instrução SELECT composto em que escape automático de campos pode quebrá-las.

```
$this->db->select('(SELECT SUM(pagamentos.quantidade) FROM pagamentos WHERE pagamentos.fatura_id=
4') AS quantidade_paga', FALSE);
```

$this->db->select_max()

Grava um Select MAX (campo) parte para a sua consulta . Você pode, opcionalmente, incluir um segundo parâmetro para renomear o campo resultante.

```
$this->db->select_max('idade');
$query = $this->db->get('membros');
// Seleciona membro com maior idade

$this->db->select_max('idade, membro_idade');
$query = $this->db->get('membros');
// Seleciona o membro com maior idade e irá renomeá-lo par membro_idade
```

$this->db->select_min()

Grava um “SELECT MIN (campo)” parte para a sua consulta . Tal como acontece com select_max (), você pode opcionalmente incluir um segundo parâmetro para renomear o campo resultante.

```
$this->db->select_min('idade');
$query = $this->db->get('membros');
// Seleciona membro com menor idade
```

$this->db->select_avg()

Grava um “SELECT AVG (campo)” parte para a sua consulta . Tal como acontece com select_max (), você pode opcionalmente incluir um segundo parâmetro para renomear o campo resultante.

```
$this->db->select_avg('idade');
$query = $this->db->get('membros');
// Retorna média das idades dos membros
```

$this->db->select_sum()

Grava um “SELECT SUM (campo)” parte para a sua consulta . Tal como acontece com select_max (), você pode opcionalmente incluir um segundo parâmetro para renomear o campo resultante.

```
$this->db->select_sum('idade');
$query = $this->db->get('membros');
// Retorna soma das idades dos membros
```

$this->db->from()

Permite-lhe para escrever a partir da porção de sua consulta :

```
$this->db->select('titulo, conteudo, data');
$this->db->from('minhaTabela');
$query = $this->db->get();
// "SELECT titulo, conteudo, data FROM minhaTabela"
```

$this->db->join()

Permite-lhe para escrever a parte JOIN de sua consulta:

```
$this->db->select('*');
$this->db->from('blogs');
$this->db->join('comentarios', 'comentario.id =  blogs.id');
$query = $this->db->get();
// SELECT * FROM blogs JOIN comentarios ON comentarios.id = blogs.id
```

Várias chamadas de função pode ser feito se você precisar de várias associações em uma consulta.

Se você precisa de um tipo específico de se juntar a você pode especificá-lo via o terceiro parâmetro da função. As opções são: esquerda, direita, externa, interna, externa esquerda e para a direita exterior.

```
$this->db->join('comentarios', 'comentarios.id =  blogs.id', 'left');
// LEFT JOIN comentarios ON comentarios.id = blogs.id
```

$this->db->where()

Esta função permite que você defina ONDE cláusulas usando um dos quatro métodos:

```
$this->db->where('nome', $nome);
// WHERE nome = 'Aleck Yann'
```

Observe que o sinal de igual é adicionado para você.

Se você usar várias chamadas de função que eles vão ser encadeados com AND entre eles:

```php
$this->db->where('nome', $nome);
$this->db->where('title', $titulo);
$this->db->where('status', $status);
// WHERE nome = 'Aleck Yann' AND titulo = 'MIT' AND status = 'ativo'
```

Pode incluir um operador, em primeiro parâmetro, a fim de controlar a comparação:

```php
$this->db->where('nome =!', $nome);
$this->db->where('idade <', $idade);
// WHERE nome != 'Aleck Yann' AND idade < 22
```

> Todos os valores passados para esta função são escapados automaticamente, produzindo querys mais seguras.

$this_db_or_where()

Esta função é idêntica à descrita acima, excepto que as várias instâncias são unidas por OR:

```php
$this->db->where('nome =!', $nome);
$this->db->or_where('idade >', $idade);
// WHERE nome != 'Aleck Yann' OR idade > 22
```

$this->db->where_in()

Gera um campo WHERE IN (‘item’, ‘item’) consulta SQL juntou-se com e, se necessário:

```php
$nomes = array('Aleck Yann', 'Jonas', 'José');
$this->db->where_in('usuarios', $momes);
// WHERE username IN ('Frank', 'Todd', 'James')
```

$this->db->where_not_in()

Gera um campo WHERE IN (‘item’, ‘item’) SQL consulta unidos com ou se for o caso:

```php
$nomes = array('Aleck Yann', 'Jonas', 'José');
$this->db->where_not_in('usuarios', $nomes);
// OR usuarios IN ('Aleck Yann', 'Jonas', 'José')
```

$this->db->or_where_not_in()

Gera um campo WHERE NOT IN (‘item’, ‘item’) SQL consulta unidos com ou se for o caso:

```php
$nomes = array('Aleck Yann', 'Jonas', 'José');
$this->db->or_where_not_in('usuarios', $nomes);
// OR usuarios NOT IN ('Aleck Yann', 'Jonas', 'José')
```

$this->db->like()

Este método permite gerar cláusulas LIKE, úteis para fazer buscas.

> Todos os valores passados para este método são escapados automaticamente.

```
$this->db->like('titulo', 'saida');
// WHERE titulo LIKE %saida% ESCAPE '!'
```

Se você usar vários chamadas de método que vai ser encadeados com AND entre eles:

```
$this->db->like('titulo', 'saida');
$this->db->like('texto', 'saida');
//WHERE titulo LIKE %saida% ESCAPE '!' AND texto LIKE %saida% ESCAPE '!'
```

$this->db->or_like()

Este método é idêntico ao descrito acima, excepto que as várias instâncias são unidas por OR:

```
$this->db->or_like ('corpo', $match );
//OR corpo LIKE %match% ESCAPE '!'
```

$this->db->not_like()

Este método é idêntico ao not_like () , exceto que as várias instâncias são unidas por OR:

```
$this->db->not_like( 'titulo' , 'match' );
// WHERE titulo NOT LIKE '%match% ESCAPE '!'
```

$this->db->or_not_like()

Este método é idêntico ao not_like () , exceto que as várias instâncias são unidas por OR:

```
$this->db->or_not_like( 'texto' , 'match' );
// OR texto NOT LIKE '%match%' ESCAPE '!'
```

$this->db->group_by()

Permite-lhe para escrever o GROUP BY parte da sua consulta :

```
$this->db->group_by( "titulo" );
// GROUP BY titulo
```

$this->db->distinct()

Adiciona a palavra-chave “DISTINCT” a um consulta:

```
$this->db->distinct();
```

```
$this->db->get( 'minhaTabela' );
// SELECT DISTINCT * FROM minhaTabela
```

$this->db->having()

Permite-lhe para escrever a parte que tem de sua consulta . Há 2 possíveis sintaxes, um argumento ou 2:

```
$this->db->having( 'usuario_id = 45' );
// HAVING usuario_id = 45
```

$this->db->or_having()

Idêntico ao having(), apenas separa várias cláusulas com “OR”:

```
$this->db->having( 'usuario_id = 45' );
// OR HAVING usuario_id = 45
```