

CodeIgniter v4 Entity

codeigniter

- https://codeigniter.com/user_guide/models/model.html#save

crear

```
namespace App\Entities;  
  
use CodeIgniter\Entity\Entity;  
  
class User extends Entity  
{  
    // ...  
}
```

usar

```
namespace App\Models;  
  
use CodeIgniter\Model;  
  
class UserModel extends Model  
{  
    protected $table          = 'users';  
    protected $allowedFields = [  
        'username', 'email', 'password',  
    ];  
    protected $returnType     = \App\Entities\User::class;  
    protected $useTimestamps = true;  
}
```

- save:

```
$user = $userModel->find($id);  
  
// Display  
echo $user->username;  
echo $user->email;  
  
// Updating  
unset($user->username);  
  
if (! isset($user->username)) {  
    $user->username = 'something new';  
}  
  
$userModel->save($user);  
  
// Create  
$user = new \App\Entities\User();
```

```
$user->username = 'foo';
$user->email     = 'foo@example.com';
$userModel->save($user);
```

- fill:

```
$data = $this->request->getPost();

# 1
$user = new \App\Entities\User();
$user->fill($data);
$userModel->save($user);

# 2
$user = new \App\Entities\User($data);
$userModel->save($user);
```

- ocultar atributos:

```
namespace App\Entities;

use CodeIgniter\Entity\Entity;

class User extends Entity
{
    protected $datamap = [
        '_role' => '_role',
    ];

    protected $attributes = [
        '___secure' => '0n',
        '_role'      => 'user',
        'about'      => '',
    ];
}

$user = new User(['___secure' => 'Off', 'about' => 'Hi, I am John!', '_role' => 'admin']);

echo 'Secure: ' . $user->___secure;
print_r($user->toArray());
print_r($user->toRawArray());

/**
 * Output:
 *
 * Secure: Off
 * Array
 * (
 *     [about] => Hi, I am John!
 *     [_role] => admin
 * )
 * Array
 * (
 *     [___secure] => Off
 *     [_role] => admin
 *     [about] => Hi, I am John!
```

```
* )
*/
```

- \$datamap: hace los atributos invisibles (uso de _) visibles
 - renombra propiedades por si ha habido cambios en la tabla (y no cambiar la lógica de la aplicación):

```
namespace App\Entities;

use CodeIgniter\Entity\Entity;

class User extends Entity
{
    protected $attributes = [
        'id'          => null,
        'full_name'   => null, // In the $attributes, the key is the db
column name
        'email'       => null,
        'password'    => null,
        'created_at'  => null,
        'updated_at'  => null,
    ];

    protected $datamap = [
        // property_name => db_column_name
        'name' => 'full_name',
    ];
}
```

- toArray(): no incluye atributos invisibles
- toRawArray(): lo incluye todo
- añadiendo lógica:

```
namespace App\Entities;

use CodeIgniter\Entity\Entity;
use CodeIgniter\I18n\Time;

class User extends Entity
{
    public function setPassword(string $pass)
    {
        $this->attributes['password'] = password_hash($pass, PASSWORD_BCRYPT);

        return $this;
    }

    public function setCreatedAt(string $dateString)
    {
        $this->attributes['created_at'] = new Time($dateString, 'UTC');

        return $this;
    }

    public function getCreatedAt(string $format = 'Y-m-d H:i:s')
    {

```

```
// Convert to CodeIgniter\I18n\Time object
$this->attributes['created_at'] =
$this->mutateDate($this->attributes['created_at']);

$timezone = $this->timezone ?? app_timezone();

$this->attributes['created_at']->setTimezone($timezone);

return $this->attributes['created_at']->format($format);
    }
}
```

- para acceder a un atributo usando funciones con lógica, se ha de respetar el formato **set** o **get** + **variable** en PascalCase. Cuando hagamos una llamada al atributo, se llamarán automáticamente.
- si hemos «mapeado» en \$datamap un atributo renombrado de la columna de la BDD, los setters y getters de ese campo han de ser los de la base de datos, no los del «alias».

From:

<https://matebcn.eu/> - miguel angel torres egea

Permanent link:

<https://matebcn.eu/development:php.codeigniter:v4:entity>

Last update: **08/06/2026 15:28**

