CodeIgniter v4 Model

codeigniter

- conexión automática a BDD
- métodos CRUD básicos
- permite extender de otros modelos

instanciar

```php
// Create a new class manually.
$userModel = new \App\Models\UserModel();

// Create a shared instance of the model.
$userModel = model('UserModel');
// or
$userModel = model('App\Models\UserModel');
// or
$userModel = model(\App\Models\UserModel::class);

// Create a new class with the model() function.
$userModel = model('UserModel', false);

// Create shared instance with a supplied database connection.
$db        = db_connect('custom');
$userModel = model('UserModel', true, $db);
```

propiedades built-in

- `$DBGRoup`: conexión alternativa a la BDD en lugar de la establecida por defecto.connecting
- `$table`: tabla primaria, solo aplica en los métodos básicos de CRUD
- `$primaryKey`
- `$useAutoIncrement = true`: poner a **false** si se ha delegado en la BDD
- `$returnType = 'array'`: retorno de `find()`, array o object
- `$useSoftDeletes = true;`: si cierto, actualiza el campo **deleted_at**
    - `withDeleted()`
- `$allowedFields`: campos permitidos en los métodos básicos CRUD
    - `save()`,`insert()`,`update()`
    - la `$primaryKey` no ha de estar en esta lista
- `$allowEmptyInserts`: Si **true** lanzará excepción en caso de insert vacío
- `$updateOnlyChanged`: Si **true** lanzará excepción en caso de que no haya nada que actualizar
- `$casts`: forzar un cast de variables después de recuperar de la BDD
    - https://codeigniter.com/user_guide/models/model.html#model-field-casting
    - no usar en conjunto con CodeIgniter v4 Entity
- Dates:
    - `$useTimestamps = false;`
    - `$dateFormat = 'datetime';`: valores permitidos: **datetime**, **date**, **int**
    - `$createdField = 'created_at';`
    - `$updatedField = 'updated_at';`
    - `$deletedField = 'deleted_at';`
- Validacions:
    - `$validationRules = [];`

        - https://codeigniter.com/user_guide/libraries/validation.html#validation-array
        - https://codeigniter.com/user_guide/models/model.html#model-setting-validation-rules
    - `$validationMessages = [];`
        - https://codeigniter.com/user_guide/libraries/validation.html#validation-custom-errors
    - `$skipValidation = false;`
    - `$cleanValidationRules = true;`
        - `cleanRules()`
        - ???
- Callbacks
    - `$allowCallbacks = true;`
    - `$beforeInsert = [];`
    - `$afterInsert = [];`
    - `$beforeInsertBatch = [];`
    - `$afterInsertBatch = [];`
    - `$beforeUpdate = [];`
    - `$afterUpdate = [];`
    - `$beforeUpdateBatch = [];`
    - `$afterUpdateBatch = [];`
    - `$beforeFind = [];`
    - `$afterFind = [];`
    - `$beforeDelete = [];`
    - `$afterDelete = [];`

casts

- ```
  protected array $casts = [
      'id'        => 'int',
      'birthdate' => '?datetime',
      'hobbies'   => 'json-array',
      'active'    => 'int-bool',
  ;
  ```
- Data Types: https://codeigniter.com/user_guide/models/model.html#data-types
    - **csv** usa `implode()` y `explode()`
    - **datetime**: formato configurado en el campo **dateFormat** de https://codeigniter.com/user_guide/database/configuration.html#database-config-explanation-of-values
- Un **?** delante del campo indica que puede ser **NULL**
- https://codeigniter.com/user_guide/models/model.html#custom-casting

métodos built-in

- `initialize()`: se ejecuta después del constructor para personalizar.initialize
- `find($primaryKey | [$primaryKey,$primaryKey,…])`:
    - retorna la fila (o filas) de las primaryKey indicadas (array si más de una)
    - si no se especifica valor, retorna todas las filas.
    - `findColumn()`: retorna los valores de una coliumna
    - `findAll()`: retorna todos los valores (si hay criterios restrictivos, los aplica)
        - `findAll($limit,$offset)`
    - `first()`: solo retorna el primer registro
    - `withDeleted()`
    - `asArray()` `asObject()` ← Entity
- `insert($data[,$bool])`
  ```

    - $data: array de valores de campos
    - $bool: si **false**, retorna un bool con el resultado de la operación
    - `getInsertID()`
    - `allowEmptyInserts()`https://codeigniter.com/user_guide/models/model.html#model-allow-empty-inserts
    - `insertBatch()`
        - $useAutoIncrement = false
- `update($primaryKey,$data)`
    - permite actualizar varias filas a la vez https://codeigniter.com/user_guide/models/model.html#update
- `save($data)`: decide si hacer insert() o update() en función de si el array contiene un campo equivalente a la de la $primaryKey
    - instancias de `Time` se convierten en cadenas con el formato definido en dateFormat['datetime'],dateFormat['date'] dentro de https://codeigniter.com/user_guide/database/configuration.html#database-config-explanation-of-values
- los métodos `save` comprueban la validez de la $primaryKey
    - https://codeigniter.com/user_guide/models/model.html#primary-key-validation
    - `validateID()`
- `delete($primaryKey,[$primaryKey,$primaryKey,…])`
    - con `$useSoftDeletes=true`, se actualiza la columna indicada en `$deletedField`
    - `purgeDeleted()`
- `chunk()`: para procesar un gran volumen de datos de manera fraccionada

in-Model Validation

- verificar antes de guardar que cumple los requisitos indicados
- usa la propiedad `$validationRules` (y complementa con `$validationMessages`:

```
    protected $validationRules = [
        'username'     =>
'required|max_length[30]|alpha_numeric_space|min_length[3]',
        'email'        =>
'required|max_length[254]|valid_email|is_unique[users.email]',
        'password'     => 'required|max_length[255]|min_length[8]',
        'pass_confirm' =>
'required_with[password]|max_length[255]|matches[password]',
    ];
    protected $validationMessages = [
        'email' => [
            'is_unique' => 'Sorry. That email has already been taken. Please
choose another.',
        ],
    ];
```

- `setValidationRule($campo,$reglas)` y su complemento `setValidationMessage($campo,$array)`
- `setValidationRules([$campo⇒$reglas,$campo⇒$reglas])` y su complemente `setValidationMessages([$campo⇒$array])`
- si no valida, el Modelo retorna **false**. Se pueden recuperar los errores con el método `errors()`
- `getValidationRules($options)`
    - $options: array con 2 claves, **except** o **only** para recoger las reglas de validación indicadas
- también se puede indicar un grupo de reglas de validaciones https://codeigniter.com/user_guide/libraries/validation.html#saving-validation-rules-to-config-file

placeholders

- https://codeigniter.com/user_guide/libraries/validation.html#rules-for-general-use
- https://codeigniter.com/user_guide/libraries/validation.html#validation-available-rules
- `is_unique`
    - `is_unique[tabla.campo,id,{id}` único campo para el id={id} (si es el campo $primaryKey)
- `alpha*`, `string`
- `field_exists`
- `in_list`, `not_in_list`
- `max_length`, `min_length`, `exact_length`
- `permit_empty`
- `required`,`required_with`,`required_without`
- `valid_email`,`valid_emails`
- `valid_date`
- `valid_json`

protecting Fields

- `$allowedFields`
    - `protect(true|false)`

Callbacks

- https://codeigniter.com/user_guide/models/model.html#defining-callbacks
- `allowCallbacks(false)`

crear

```
namespace App\Models;

use CodeIgniter\Model;

class UserModel extends Model
{
    // ...
}
```

From:
https://matebcn.eu/ - **miguel angel torres egea**

Permanent link:
https://matebcn.eu/development:php:codeigniter:v4:model

Last update: **08/06/2026 10:31**