Codeigniter i les API

Operation	Method	Controller Route	Presenter Route	Controller Function	Presenter Function
New	GET	photos/new	photos/new	`new()`	`new()`
Create	POST	photos	photos	`create()`	`create()`
Create (alias)	POST		photos/create		`create()`
List	GET	photos	photos	`index()`	`index()`
Show	GET	photos/(:segment)	photos/(:segment)	`show($id = null)`	`show($id = null)`
Show (alias)	GET		photos/show/(:segment)		`show($id = null)`
Edit	GET	photos/(:segment)/edit	photos/edit/(:segment)	`edit($id = null)`	`edit($id = null)`
Update	PUT/PATCH	photos/(:segment)		`update($id = null)`	
Update (websafe)	POST	photos/(:segment)	photos/update/(:segment)	`update($id = null)`	`update($id = null)`
Remove	GET		photos/remove/(:segment)		`remove($id = null)`
Delete	DELETE	photos/(:segment)		`delete($id = null)`	
Delete (websafe)	POST		photos/delete/(:segment)	`delete($id = null)`	`delete($id = null)`

Codeigniter permet crear facilment RESTful APIs per als recursos del nostre projecte.
Codeigniter disposa de les classes següents:

- ResourceController
- PresenterController

API Request

Podem saber quin tipus de petició tenim, si es tracta d'una petició tipus API emprant:

```
// Check for AJAX request.
if ($request->isAJAX()) {
    // ...
}
```

Això és possible si està establert el camp header X-Requested-With

En aquest punt si les dades ens arriben via JSON, en aquest cas a la capçalera del request el camp CONTENT_TYPE tindria el valor de “application/json”. Per obtenir els parametres que ens arriben a la API via body emprarem les funcions següents:

- **$json = $request->getJSON();**
- **$json = $request->getVar();**

En cas de voler obtenir un parametre concret de la petició podriem emprar:

- gerJsonVar('param');
- getVar('param');

Exemple

```
// With a request body of:
/*
{
    "foo": "bar",
    "fizz": {
        "buzz": "baz"
    }
}
*/

$data = $request->getVar('foo');
// $data = "bar"
$data = $request->getVar('fizz.buzz');

// With the same request as above
$data = $request->getJsonVar('fizz');
// $data->buzz = "baz"

$data = $request->getJsonVar('fizz', true);
// $data = ["buzz" => "baz"]
```

Controller response

Podem accedir a la resposta que s'envia al navegador emprant el propi controlador, via:

$this->response->

Dins l'objecte resposta podrem establir tant els elements de capçalera si ens fan falta, com el cos de la mateixa.

$this->response->setStatusCode(404)->setBody('Nope. Not here.');

$this->response->setStatusCode(404, 'Nope. Not here.');

També es possible enviar la resposta a client, en format JSON o XML.

```
$data = [
    'success' => true,
    'id' => 123,
];
return $this->response->setJSON($data);
// or
return $this->response->setXML($data);
```

Per establir altres valors de capçalera podrem emprar la funció setHeader()

$this->response->setHeader('Location', 'http://example.com')

->setHeader('WWW-Authenticate', 'Negotiate');

Podem afegir elements a una capçalera emprant appendHeader

$this->response->setHeader('Cache-Control', 'no-cache')

->appendHeader('Cache-Control', 'must-revalidate');

En cas necessari podem eliminar un item de capçalera via removeHeader

$this->response->removeHeader('Location');

API Response

CI4 proporciona un objecte anomenat API Response trait que permet a qualsevol controlador retornar respostes simples amb els valors HTTP status segons ens interessi.

```
namespace App\Controllers;

use CodeIgniter\API\ResponseTrait;

class Users extends \CodeIgniter\Controller
{
    use ResponseTrait;

    public function index(){
        //...
        // Respond with 201 status code
        return $this->respondCreated();
    }
}
```

Aquest objecte ens permetria retornar els següents status code:

```
// Generic response method
$this->respond($data, 200);
// Generic failure response
$this->fail($errors, 400);
// Item created response
$this->respondCreated($data);
// Item successfully deleted
$this->respondDeleted($data);
// Command executed by no response required
$this->respondNoContent($message);
// Client isn't authorized
$this->failUnauthorized($description);
// Forbidden action
$this->failForbidden($description);
// Resource Not Found
$this->failNotFound($description);
// Data did not validate
$this->failValidationError($description);
// Resource already exists
$this->failResourceExists($description);
// Resource previously deleted
$this->failResourceGone($description);
// Client made too many requests
$this->failTooManyRequests($description);
```

Revisió núm. 3
★Admin l'ha creat 2025-06-06 16:03:26 UTC
Admin l'ha actualitzat 2025-06-06 16:22:43 UTC