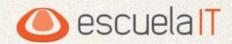




Desarrollo de aplicaciones web con Laravel 5

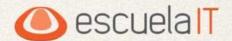
Clase 4. Request, R. Validations, Responses y Cookies Carlos Ruiz Ruso · @micromante





Ciclo de vida

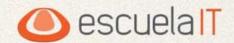
http://laravel.com/docs/5.0/lifecycle

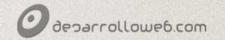




Ciclo de vida de una petición

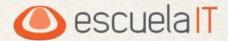
- 1. Las peticiones entran por public/index.php
- 2. bootstrap/start.php crea la app y detecta los entornos
- 3. app/Http/Kernel.php
- 4. Middlewares
- 5. Service providers
- 6. Routes
- 7. Controller ... models, helpers...
- 8. Respuesta al cliente

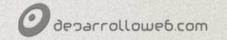




Requests

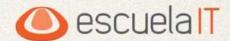
Peticiones

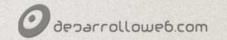




HTTP Request

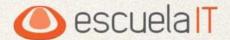
- La clase Request provee de métodos muy útiles para trabajar las peticiones HTTP.
- Viene por defecto
- Heredamos la potencia de Request de Symfony
- La sintaxis es clara y sencilla
- Dos vías de trabajo
 - Via Facade
 - Via Inyección de dependencias





Request via Facade

- Simplemente
 - \$id = Request::input('id');
- Si estás trabajando con namespaces debes añadir use Request; arriba de la clase.





Request via Dependency Injection

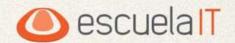
En el método de la clase

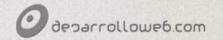
public function almacenar(Request \$request) { ... }

Globalmente en la clase en un atributo de clase

```
public function __construct(Request $request) {
    $this->request = $request;
}
```

^{*}Luego mediante \$this->request podemos hacer uso de la misma.





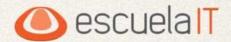
Inputs





DEMOSTRACIÓN

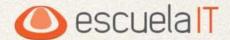
con POSTMAN

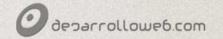




INPUTI

- Todos los valores
 - Request::all();
- Algunos valores
 - Request::only('name', 'password'); //Solamente indicados
 - Request::except('dni'); //Todos menos dni
- Valores por GET, POST... No importa el verbo
 - Request::input('name');
 - Request::input('product.name'); //Si el form tiene un array

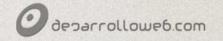




INPUTII

- Existe el valor
 - Request::has('name')
- Si trabajamos con librerías javascript como angular o backbone puede ser que los datos del input se envíen como json, trabajamos con estos datos con Request::all() o Request::input(...) ya que laravel hacer la conversión.
- Cuando hacemos redirect los datos se borran excepto si hacemos.
 - return redirect('testcontroller')->withInput();





INPUT III

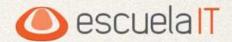
- Con ficheros, siempre es el valor del campo name del form.
 - Request::file('imagen');
 - Request::file('imagen')->move('path');
 - Request::hasFile('imagen');
 - ->getSize();
 - ->getMimeType();
 - ->getClientOriginalExtension();
 - ->getClientOriginalName();
 - ->getRealPath();
- Otros métodos: http://api.symfony.com/2.
 5/Symfony/Component/HttpFoundation/File/UploadedFile.html

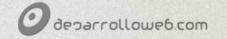




Extra

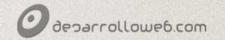
- Obtener la URI de la llamada
 - \$uri = Request::path();
- Determinar si es una llamada AJAX
 - Request::ajax();
- Obtener el método
 - Request::method()
 - Request::isMethod('post')
- URL actual
 - \$url = Request::url();
- Flash data
 - Request::flash() ← evitamos perder la request entre saltos next o controladores.





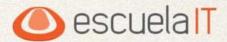
Request Validation

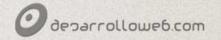




Validaciones

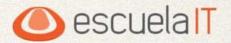
- La validación de Laravel 4 crece en Laravel 5 con la validación a nivel Request
- Ahora podemos hacer validación de la Request con los mismos metodos.
- Reglas de validación
 - http://laravel.com/docs/5.0/validation#available-validation-rules

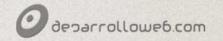




Validaciones Controller

```
/**
* Store the incoming blog post.
* @param Request $request
* @return Response
public function store(Request $request)
   $this->validate($request, [
        'title' => 'required|unique|max:255',
        'body' => 'required',
   ]);
```

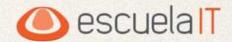


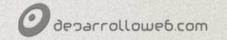


Ejemplo práctico de validación Request

```
namespace App\Http\Requests\Frontend\User;
use App\Http\Requests\Request;
class UpdateProfileRequest extends Request {
      public function authorize()
            return true;
                                                                                     <?php namespace App\Http\Controllers\Frontend;</pre>
     public function rules()
                                                                                     use App\Http\Controllers\Controller;
                                                                                     use App\Repositories\Frontend\User\UserContract;
use App\Http\Requests\Frontend\User\UpdateProfileRequest;
            return [
                              => 'required',
                   'email' -> 'sometimes|required|email',
                                                                                     class ProfileController extends Controller {
           1;
                                                                                          public function edit($id) {
                                                                                              return view('frontend.user.profile.edit')
                                                                                                  ->withUser(auth()->user($id));
                                                                                         public function update($id, UserContract $user, UpdateProfileRequest $request) {
    $user->updateProfile($id, $request->all());
    return redirect()->route('frontend.dashboard')->withFlashSuccess("Profile successfully updated.");
```

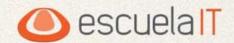
Veremos esta parte en profundidad en próximos temas





Responses & Redirects

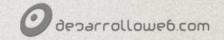
respuestas y redirecciones



Redirects

Responder con una redirección



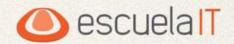


Redirects

return redirect('login') 0 return redirect('user')->with('message', 'Algo ha pasado'); 0 return redirect()->back(); 0 return redirect()->back()->withInput(); 0 return redirect()->route('login'); 0 return redirect()->route('login', [\$user]); 0 return redirect()->route('login', ['user' => 1]); //login/{user_id} 0 return redirect('user')->withInput(); 0 return redirect('user')->withInput(Request::except('password')); 0 return redirect()->action('App\Http\Controller\HomeController@method'); 0 return redirect()->action('App\Http\Controller\HomeController@method',['user' => 1); 0

return redirect()->action('App\Http\Controller\HomeController@method',[1]);

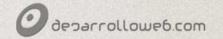
0





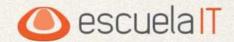
Responses





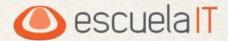
Responses

- Json y JsonP
 - o return response()->json(array(...));
 - return response()->json(array(...))->setCallback(\$request->input('callback');
- Download
 - return response()->download(\$path, \$name, \$headers);



Cookies

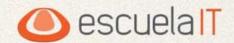
http://laravel.com/docs/5.0/requests#cookies





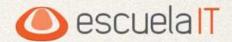
Cookies

- Obtener cookie
 - \$value = Request::cookie('name');
- Añadir cookie a la respuesta
 - \$\square \text{\$\text{response} = new Illuminate\Http\Response('Hello World');} \square \text{\$\text{response->withCookie(cookie('name', 'value', \text{\$\text{minutes}));}} \end{array}\$
- Cookies para siempre
 - \$response->withCookie(cookie()->forever('name', 'value'));



Helpers

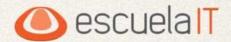
http://laravel.com/docs/5.0/helpers





¡Ahora os toca a vosotros!

Practicar lo aprendido que hay bastante materia





¡MUCHAS GRACIAS A TODOS POR VUESTRA PARTICIPACIÓN!

Aquí me teneis para lo que os pueda ayudar.
Estais invitados a seguirme en la redes sociales como @micromante o Carlos Ruiz Ruso
www.micromante.com