

# Desarrollo de aplicaciones web con Laravel 5.1

**Clase 10. Seguridad, Cache, idiomas, colecciones... y buenas prácticas**  
Carlos Ruiz Ruso • @micromante

# Seguridad

# Hashing

- Potente fachada para manejo de hash
- Guardado de claves Bcrypt
  - Potente contra ataques fuerza bruta
  - Única dirección de encriptación
  - Siempre comparamos la clave del usuario con la bd encriptando de igual forma.
  - Algoritmo Blowfish con salto y ampliación
  - Novedad en PHP 5.5
  - Tienen siempre el mismo tiempo de decodificación así se puede proteger con timing attacks
  - <http://librosweb.es/tutorial/la-nueva-api-para-codificar-contrasenas-de-php-55/>
- Usado en sistemas Auth de laravel 5



# Hashing

- `$password = Hash::make('clave');`
- `$password = bcrypt('clave');`
- `if(Hash::check('clave',$hasheddb) ) { ... }`
- `if(Hash::needsRash($hashed)){ ... }`

# Hashing en Update

```
public function updatePassword(Request $request, $id)
{
    $user = User::findOrFail($id);

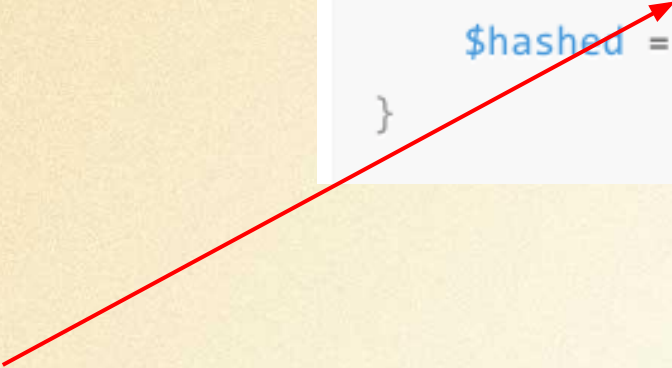
    // Validate the new password length...

    $user->fill([
        'password' => Hash::make($request->newPassword)
    ]->save());
}
```



## Si cambia el factor por tiempo

```
if (Hash::needsRehash($hashed)) {  
    $hashed = Hash::make('plain-text');  
}
```





# Seguridad CSRF en AJAX I

```
$.ajaxSetup({  
    headers: {  
        'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')  
    }  
});
```



**VerifyCsrfToken** Middleware busca X-CSRF-TOKEN en llamadas Ajax

# Seguridad CSRF en AJAX II

{!! csrf\_field() !!}

```
<meta name="csrf-token" content="" />

var CSRF_TOKEN = $('meta[name="csrf-token"]').attr('content');

$.ajax({
  url: '/home/upload/',
  type: 'POST',
  data: {_token: CSRF_TOKEN},
  dataType: 'JSON',
  success: function (data) {
    console.log(data);
  }
});
```



# Cacheo de Rutas

Incrementa la velocidad de respuesta su uso.

No ejecutar en dev ya que podemos no ver los cambios de routes.php

```
php artisan route:cache  
php artisan route:clear
```

# Traducciones

# Traducciones

- Sencillo sistema para las traducciones
- `resources/lang/{locale}`
- `App::setLocale($lang);`
- `'fallback_locale'` → idioma cuando no encuentra una cadena
- `trans('filename.linename')` ← debemos tener el fichero y la linea en el array.
- `trans('filename.linename',['nombre' => 'Carlos'])` → `Hola :nombre`
- Pluralización
  - <http://laravel.com/docs/5.1/localization#pluralization>



# Colecciones

# Colecciones

- **Wrapper para trabajar con arrays**
- **Operaciones sobre los elementos muy rapidas**
- **`$collection = collect(['Carlos','Luis','Miguel']);`**
- **Existen muchos métodos para trabajar con ellos.**
- **Veamos algunos...**

# Colecciones

- **Chunk** ← particiones en grupos

```
@foreach ($products->chunk(3) as $chunk)
    <div class="row">
        @foreach ($chunk as $product)
            <div class="col-xs-4">{{ $product->name }}</div>
        @endforeach
    </div>
@endforeach
```





# Colapsado

- **Collapse** ← agrupar en un solo nivel

```
$collection = collect([[1, 2, 3], [4, 5, 6], [7, 8, 9]]);

$collapsed = $collection->collapse();

$collapsed->all();

// [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

# Diferencia

- **diff** ← Diferencia entre 2 colecciones

```
$collection = collect([1, 2, 3, 4, 5]);  
  
$diff = $collection->diff([2, 4, 6, 8]);  
  
$diff->all();  
  
// [1, 3, 5]
```

# Diferencia

- **each** ← Operaciones sobre elementos

```
$collection = $collection->each(function ($item, $key) {  
    //  
});
```



# Group By

```
$collection = collect([
    ['account_id' => 'account-x10', 'product' => 'Chair'],
    ['account_id' => 'account-x10', 'product' => 'Bookcase'],
    ['account_id' => 'account-x11', 'product' => 'Desk'],
]);

$grouped = $collection->groupBy('account_id');

$grouped->toArray();

/*
[
    'account-x10' => [
        ['account_id' => 'account-x10', 'product' => 'Chair'],
        ['account_id' => 'account-x10', 'product' => 'Bookcase'],
    ],
    'account-x11' => [
        ['account_id' => 'account-x11', 'product' => 'Desk'],
    ],
]
*/
```

# Mezclado

```
$collection = collect(['product_id' => 1, 'name' => 'Desk']);

$merged = $collection->merge(['price' => 100, 'discount' => false]);

$merged->all();

// ['product_id' => 1, 'name' => 'Desk', 'price' => 100, 'discount' => false]
```

# Añadir al inicio o al final

```
$collection = collect([1, 2, 3, 4, 5]);
```

```
$collection->prepend(0);
```

```
$collection->all();
```

```
// [0, 1, 2, 3, 4, 5]
```

```
$collection = collect([1, 2, 3, 4]);
```

```
$collection->push(5);
```

```
$collection->all();
```

```
// [1, 2, 3, 4, 5]
```



# Sort

```
$collection = collect([5, 3, 1, 2, 4]);  
  
$sorted = $collection->sort();  
  
$sorted->values()->all();  
  
// [1, 2, 3, 4, 5]
```

# ¡Muchos más!

<http://laravel.com/docs/5.1/collections#available-methods>

# Recursos interesantes

<http://aimeos.org/project/laravel-shop-package/>

<http://laravel.com/docs/5.1/authentication#social-authentication>

<http://laravel.com/docs/5.1/billing> (prefiero no usarlo...)

<https://github.com/cartalyst/sentry/tree/feature/laravel-5>

# Autores

[http://www.laravelbestpractices.com/#design\\_patterns](http://www.laravelbestpractices.com/#design_patterns)

<http://daylerees.com/>

<http://taylorotwell.com/>



# MUCHAS GRACIAS A TODOS!

Podéis seguirme en la redes sociales como @micromante o Carlos Ruiz Ruso  
[www.micromante.com](http://www.micromante.com)