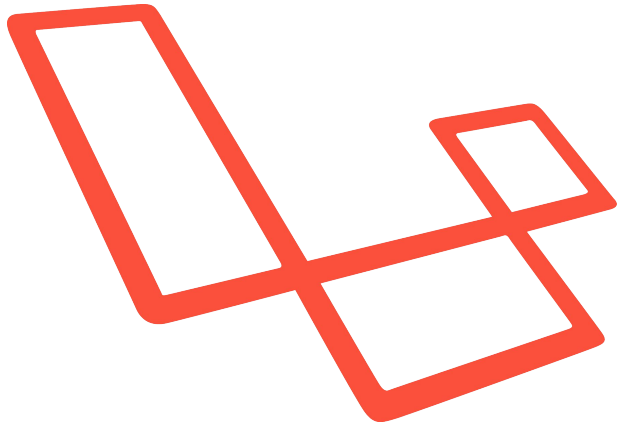
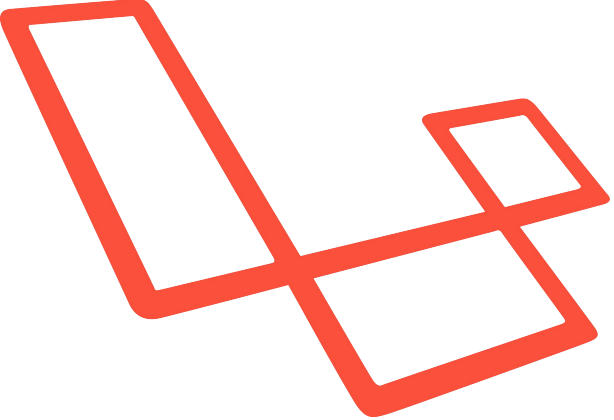


LARAVEL

Framework - Capacités - Implémentation



LARAVEL

Framework - Capacités - Implémentation

Pour le retour d'une POO en PHP

Pour sa flexibilité dans le projet

Pour ses lignes de commandes : CLI *artisan*

Pour ses objets de migrations de base

Pour son moteur de template : *Blade*

Pour son MVC aux petits oignons

Pour sa rapidité (et son poids)

Pour son routing d'URL

Pour son ORM (Object-Role Modeling) : *Eloquent*

Framework

- Ecriture et lecture simple :

```
$article->translation['fr']->title
```

- Moteur de template *Blade* :

```
<i>{{ $article->translation['fr']->title }}</i>
```

- Invite de commande *Artisan* :

```
php artisan make:model Article --plain
php artisan route:list
```

Framework

- 4 dossiers principaux :

Modèles : \App*
Controleurs : \App\Http*
Vues : \resources*
Assets : \public*

- 2 dossiers de configuration :

Principal : \.env
Autre : \config*

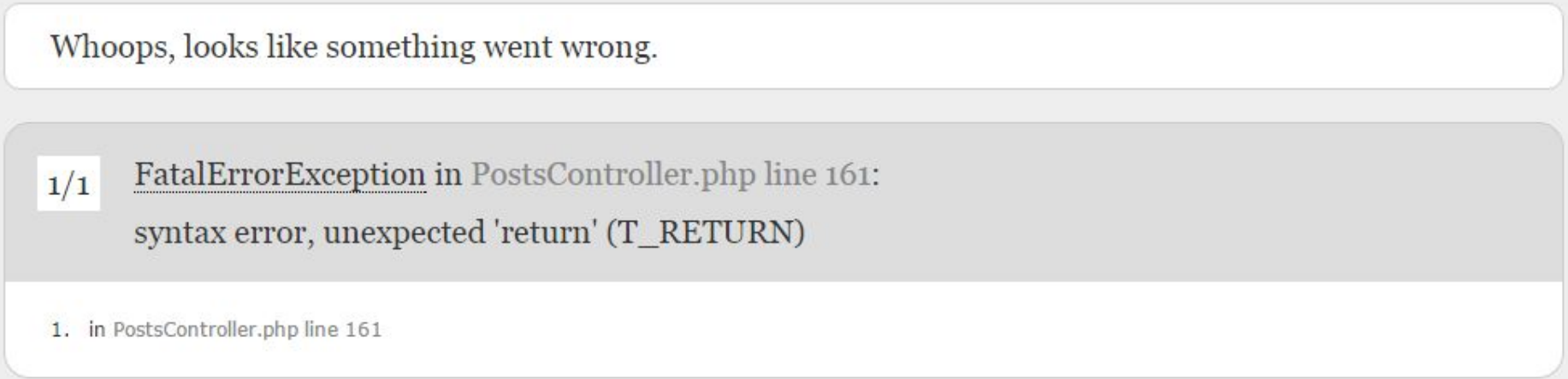
- 1 dossier pour la BDD :

Migrations : \database*

Framework

Erreurs :

Des retours d'erreurs formatés, avec toutes les infos nécessaires.
Pour voir ses merdes dans de bonnes conditions !



Framework

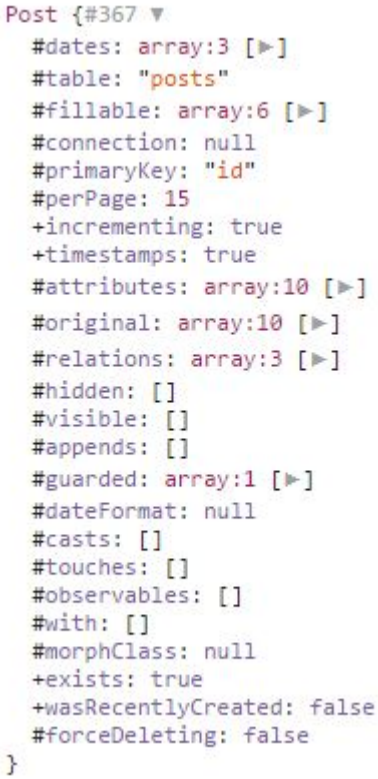
Debug :

Pour pleurer devant des tableaux illisibles ...

En plus des débugs classiques (`var_dump()` ou `print_r()`)
Laravel propose un outil de débug interne :

```
dd($article);
```

Le `dd()` correspond à un `var_dump()` suivi d'un `die()`



Framework

Namespacing :

Comme les kinders, mais en mieux !

L'un des fondements de la POO, l'encapsulation :

```
namespace App\model;

class Comment extends Model
{ [...] }
```

```
namespace App\tests;

class Comment extends Model
{ [...] }
```

Capacités(route)

Liste non exhaustive des capacités de Laravel 5 :

- Routage des vues et des controleurs dans le fichier \App\routes.php (écritures d'URL):

```
Route::resource('article', 'ArticlesController');
```

Correspond à définir les adresses

- \article
- \article\{id}
- \article\{id}\edit
- \article\create

Capacités (middleware)

- Les middlewares !

```
php artisan make:middleware Ip
```

```
class Ip {

public function handle($request, Closure $next)
    {
        if($request->ip() == "127.0.0.1"){
            return $next($request);
        }
        return response('Unauthorized','403');
    }
}
```

Capacités(controller)

- Expérience utilisateur

Différentes méthodes pour gérer les redirections, les retours d'erreurs, les sessions/cookies :

- Redirect page `return back();`
- Redirect view `return view('articles.show');`
- Erreurs `return back()->with('error','Erreur!');`

Capacités(model)

- Liaisons dans la BDD (hasMany / belongsTo) :

Une table dépend d'une autre (belongsTo)

```
public function blog()
    return $article->belongsTo('App\Blog');
```

Une table en possède une autre (hasMany)

```
public function comments()
    return $article->hasMany('App\Comment');
```

Capacités(controller)

- ORM Eloquent :

I'm an Eloquent

Construire simplement ses requètes.

```
Article::create($datas)

$article = Article::where('id','=',$id)->firstOrFail();

Article::where('id','=',$id)->with('comments')->firstOrFail();

$article>update($datas);

$article->delete();
```

Capacités(model)

- Getter / Setter :

Créer ou modifier des attributs de l'objet après l'accès en base

```
public function getThumbAttribute() {
    return "/img/articles/" .  this->id."/thumbimg.jpg";
}
```

```
public function setSlugAttribute($title) {
    return Str::slug($title);
}
```

Capacités(model)

- Event :

Dès qu'une action se produit (suppression, modification, création)

```
protected static function boot()
{
 parent::boot();

 static::deleting(function($article) {
     $article->comments()->delete();
 });
}
```

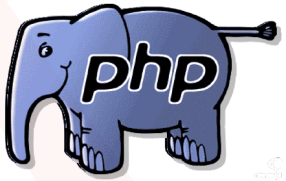
Capacités (autres)

- Gestion de différents espaces disques
- Objet de Mailer intégré
- Objet de Tests unitaires intégré
- Mise en cache automatique
- Service Providers / Façades
- Outil de localisation pour multilingue

Implémentation

Prérequis :

- Composer pour toutes les dépendances
- PHP >= 5.5.9
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension



Implémentation

Composer :

Créer le projet : `composer create-project MyName laravel/laravel`

Configuration :

Tout se trouve dans le dossier \config*

Générer la clé d'application du .env : `php artisan key:generate`

Implémentation

Via l'invite de commande

Le premier modèle :

```
php artisan make:model Article
```

Premier controleur :

```
php artisan make:controller articlesController
```

Première entrée en base :

```
php artisan make:migration maMigration
php artisan migrate
```

La première vue \resources\app.blade.php:

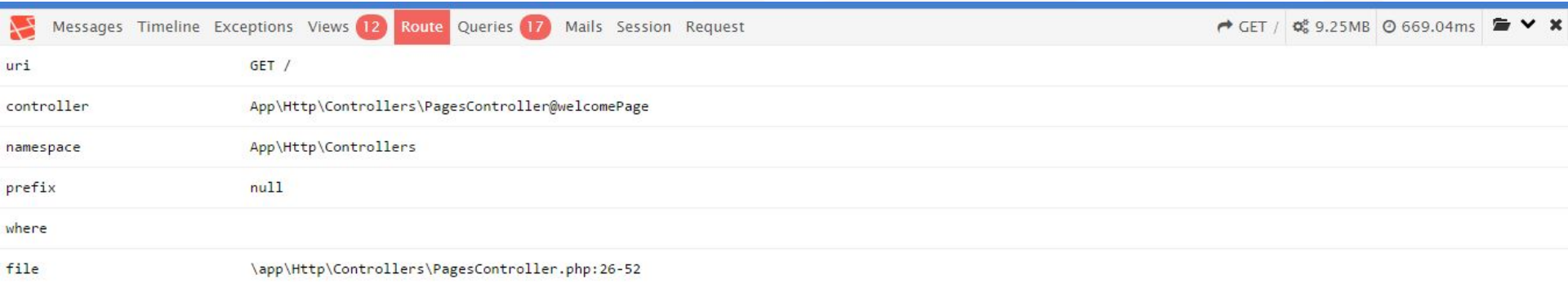
```
@include('includes.head')
<body>
@yield('content')
</body>
@include('includes.foot')
```

Implémentation

Liste de quelques modules Laravel pertinent :

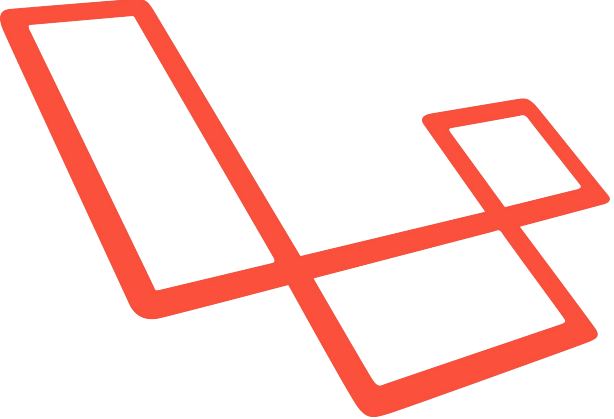
Outils non intégrés par défaut, mais qui valent le coup :

- Barryvdh Debugbar : Barre d'info (nombre de vues, de requêtes, temps d' execution, route utilisée)



Implémentation

- Intervention : Gestion d'image en PHP
- Gravatar : Avatars via … Gravatar. no joke
- Beaucoup d'autres, à trouver par vous même !



LARAVEL

A retenir

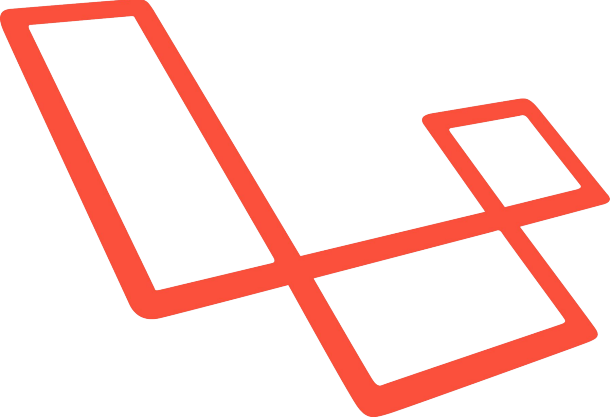
L'invite de commande Artisan pour créer des migrations, des modèles et des controleurs.

Utiliser les migrations pour tenir une BDD propre et à jour.

Optimiser ses requêtes avec Eloquent.

Respecter le MVC et ne pas hésiter à fragmenter les vues Blade pour garder le code lisible.

Lire la doc … et quelques sites utiles, comme laracast.com ou laravel.io.
Aussi la formation Laravel de Graphikart :)



Merci

Martin Villanove

Développeur Web
chez Bulko

m-vi@bulko.net
07 86 81 00 44