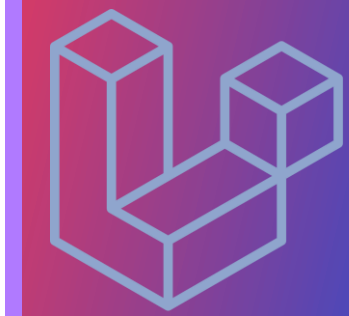
LARAVEL

Module 06
Requêtes et Validations

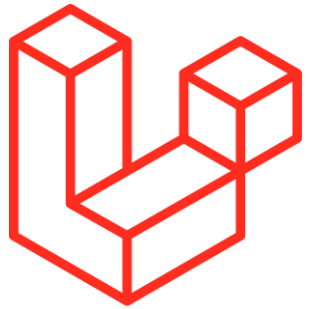
Au programme dans ce module

- Chapitre 1 : Qu'est ce que Request ?
  - Une boite à outils de la requête client
- Chapitre 2 : Les formulaires avec Laravel
  - Quelques changements
  - Rien qu'un peu de pratique
- Chapitre 3 : Qu'est ce que la Validation ?
  - Validation de formulaires
  - Beaucoup de règles
  - Quelques règles essentielles
  - // Traitement des erreurs dans Blade
  - // Insertion des valeurs en cas d'erreurs dans Blade
  - Pratiquer c'est mieux



Qu’est ce que Request ?

Chapitre 1

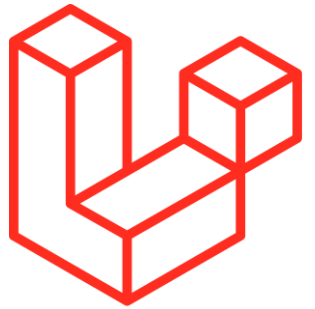


Une boite à outils de la requête client

- Une librairie « Illuminate\Http\Request » du **cœur de symfony et amélioré par Laravel**.
- **Combine** toutes les données de la requête client dont **« $_GET » et « $_POST »**.
- Des méthodes simples pour utiliser ces données comme « **$request->all()** » pour obtenir un tableau de toutes les variables de GET et POST.
- Le **helper « request() »** permet d'utiliser les données de la requête client où vous le souhaitez en faisant : **« $request = request() ».**
- Appeler une variable depuis « $request » fait chercher à Laravel la présence de cette variable dans la requête du client et permet de la récupérer : « **$request->ma_var_front** ». Si la variable n'existe dans la requête client, « null » sera retourné.
- Il est possible de différencier les fichiers uploadés dans la requête client en utilisant « **$request-> allFiles()** » qui renverra le tableau des fichiers uploadés ou « $request-> file('FILE_NAME') » pour récupérer un fichier précis. Le nom du fichier correspond à la variable choisie dans le formulaire du navigateur.
- Une **montagne d'autres fonctionnalités** présentées dans la documentation.

Les formulaires avec Laravel

Chapitre 2



Quelques changements

```blade
<form action="@route('comment.store')" method="POST">

    {{-- Préviens les attaques XSS --}}
    @csrf

    {{-- Un simple input --}}
    <input id="title" name="title" type="text">
    <label for="title">Titre</label>

    {{-- Un simple input --}}
    <input id="message" name="message" type="text">
    <label for="message">Message</label>

    <input type="submit" value=">>>">

</form>
```

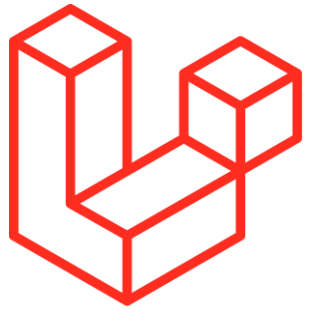
→ La directive « @csrf » et le Middleware VerifyCsrfToken.php préviennent les attaques frontend de type XSS.

Rien qu'un peu de pratique

- Créer un contrôleur « CommentsController ».
- Créer une vue Blade « comments.create » avec la route « comments/create » et la méthode « create() » dans le contrôleur CommentsController.
- Créer un formulaire HTML simple comme vous savez déjà le faire dans la vue. La méthode du formulaire à utiliser est « POST ». L'action du formulaire doit être une nouvelle route et méthode à créer, que vous nommerez « store ». La méthode peut être vide pour le moment.
- Ajouter des inputs de votre choix à votre formulaire (Trouvez-en au moins 2). N'oublier pas d'ajouter un attribut « name » à vos inputs.
- Ajouter la sécurité XSS dans le formulaire avec la directive Blade.
- Récupérer les inputs du formulaire dans la méthode « store() » avec Request ($request) comme découvert dans les modules précédant. Utiliser « dd() » pour vérifier leur contenu. Nous utiliserons ses variables dans le chapitre suivant.

Qu’est ce que la Validation ?

Chapitre 3



Validation de formulaires

```php
public function store(Request $request)
{
    // $request->validate() retourne la requête
    // si la validation ne match pas
    $request->validate(
        [
            'email' => 'required|email|max:255',
            'age'   => 'integer|max:120',
            'title' => 'required|max:255',
            'body'  => 'required|max:255',
        ],
        [
            'email.required' => 'Mon message perso',
        ]
    );
}
```

- Apporte un traitement automatisé et personnalisable pour la sécurité, le filtrage et la vérification des formulaires.
- Nombreuses règles de validation pour répondre à un grand nombre de cas.
- Création de nouvelles règles.
- Personnalisation des réponses à afficher en cas d'erreurs.

Qu'est ce que la Validation ?

Beaucoup de règles

Accepted
Active URL
After (Date)
After Or Equal (Date)
Alpha
Alpha Dash
Alpha Numeric
Array
Bail
Before (Date)
Before Or Equal (Date)
Between
Boolean
Confirmed
Date
Date Equals
Date Format
Different
Digits
Digits Between
Dimensions (Image Files)
Distinct
E-Mail
Ends With
Exclude If
Exclude Unless
Exists (Database)
File
Filled
Greater Than
Greater Than Or Equal
Image (File)
In
In Array
Integer
IP Address
JSON
Less Than
Less Than Or Equal
Max
MIME Types
MIME Type By File
Extension
Min
Not In
Not Regex
Nullable
Numeric
Password
Present
Regular Expression
Required
Required If
Required Unless
Required With
Required With All
Required Without
Required Without All
Same
Size
Sometimes
Starts With
String
Timezone
Unique (Database)
URL
UUID

- Gain de temps dans la vérification de formulaires.
- Possibilité d'appliquer des validations à plusieurs méthode de contrôleur en centralisant la validation dans des fichier de Requête comme ci-suit :
  « app/Http/Requests/…Request.php »
- Langue par défaut au choix après import/téléchargement des langues nécessaires.
- Voir : Laravel Available Validation Rules

Quelques règles essentielles

```php
$request->validate(
    [
        'email' => 'required|email|max:60', // Obligatoire + email + longeur max 60

        'password' => 'required|confirmed|min:8', // Obligatoire + confirmé + min 8

        'name'  => 'string|alpha|min:2', // Text + alphabétique + longueur minimale 2

        'age'   => 'nullable|integer|size:26', // Nullable + entier + Egale à 26

        'birth'  => 'date_format:d/m/Y', // Date au format 01/11/2033

        'image' => 'image|max:1024', // Image : jpeg, png, bmp, gif, svg, ou webp

        'certificate' => 'file|max:1024', // Fichier + Inférieur à 1024 Ko

        'document' => 'mimes:doc,docx,pdf', // Type word ou pdf (Many extension)
    ]
);
```

Installer des langues en plus

```
#> composer require --dev laravel-lang/common
```

→installation du paquage optionnel des multiples langues pour Laravel

```
#> php artisan lang:add en fr de es
```

→Ajout de fichier pré-traduit pour les langues « en », « fr », « de » et « es » dans le dossier /lang/
→Liste des locales supportées: https://laravel-lang.com/usage-list-of-locales.html#list_of_codes

```
#> php artisan lang:update
```

→Update tous les fichiers de langues en ajoutant les nouvelles key:value suite à une mise à jour

Pratiquer c'est mieux

- Réutilisez le formulaire de la vue « pokemons.create » et ajoutez les champs : « quantity », « provider_name », « provider_email » et « picture ».
- Dans la méthode qui gère l'action du formulaire, ajoutez une validation Laravel des données de la requête. Pour cela, définissez plusieurs règles de validation qui vous semblent intéressantes (sauf « unique » et « exists »). Il y en a d'autres sur la doc : https://laravel.com/docs/10.x/validation#available-validation-rules
- Comme pour les directives Blade ou les macro des Collection, il est possible de créez vos propres règles de validation : https://laravel.com/docs/10.x/validation#custom-validation-rules

Fin du module