

BTS SIO :: SLAM :: PARTIE 5

LARAVEL

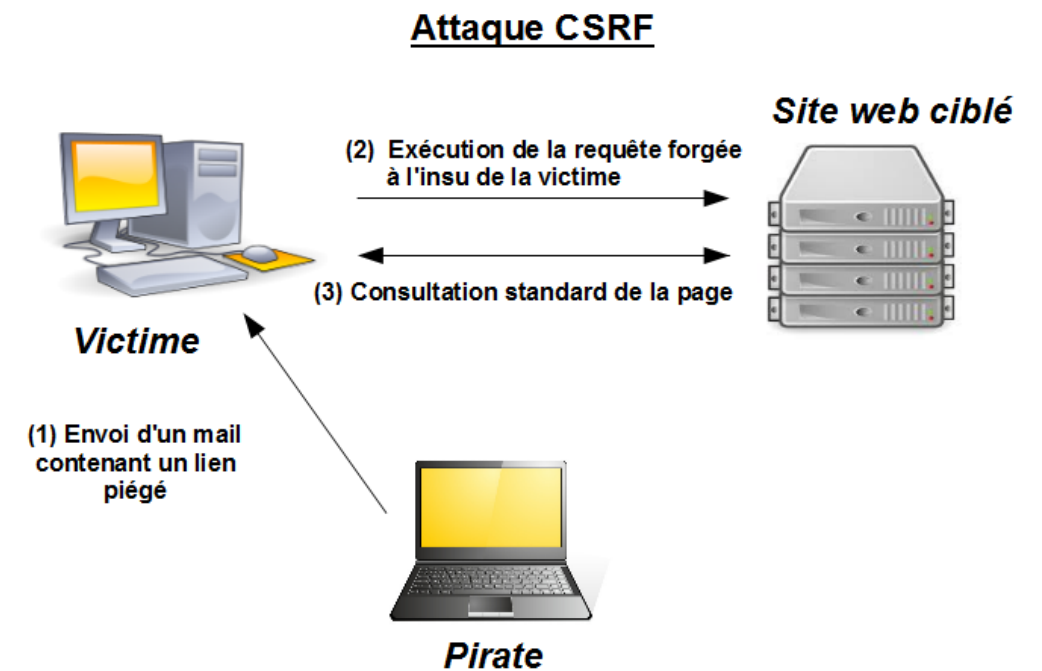
LARAVEL

LES FORMULAIRES

SECURISER LES FORMULAIRES

- ▶ Le formulaire HTML est un moyen simple pour les interactions entre un site et ses visiteurs.
- ▶ Il est donc aussi un vecteur d'attaque potentiel
- ▶ Laravel propose une protection face aux attaques de type CSRF (Cross Request Forgery)

ATTAQUES CSRF



- ▶ Le but est de transmettre une requête HTTP falsifiée à l'insu d'un utilisateur et avec ses droits.
- ▶ Si l'utilisateur est admin : l'attaquant aura tous les droits.
- ▶ Laravel génère afin d'éviter cela un jeton de sécurité.
`{{ csrf_field() }}`

CREATION D'UN FORMULAIRE : LES ROUTES

- ▶ Il faut 2 routes :
 - ▶ Une qui affiche le formulaire (retourne une vue)
 - ▶ Une pour traiter le formulaire (vérification, enregistrement, redirection, etc)

```
// Affichage du formulaire
Route::get('/articles/create', 'ArticleController@create');

// Traitement du formulaire
Route::post('/articles/create', 'ArticleController@store');
```

CREATION D'UN FORMULAIRE : LE CONTRÔLEUR

- ▶ Il existe déjà normalement avec une méthode `show()`
- ▶ Si c'est pas le cas il faut le créer avec Artisan :

`php artisan make:controller ArticleController`

- ▶ On va créer la première méthode `create()` qui se contente de retourner une vue contenant le formulaire.

```
class ArticleController extends Controller
{
    public function create()
    {
        return view('articles.create');
    }

    public function show($n)
    {
        return view('article')->with('numero', $n);
    }
}
```

CREATION D'UN FORMULAIRE : LE CONTRÔLEUR

Si on teste on voit que la route est trouvée mais il nous manque la vue.

InvalidArgumentException
View [articles.create] not found.

G ⓘ 📄 COPY

Application frames (2) All frames (57)

- 56 **InvalidArgumentException**
.../vendor/laravel/framework/src/Illuminate/View/ViewFinder.php:137
- 55 **Illuminate\View\ViewFinder findInPaths**
.../vendor/laravel/framework/src/Illuminate/View/ViewFinder.php:79
- 54 **Illuminate\View\ViewFinder find**
.../vendor/laravel/framework/src/Illuminate/View/ViewFactory.php:130
- 53 **Illuminate\View\Factory make**
.../vendor/laravel/framework/src/Illuminate/Foundation/helpers.php:1011

/Users/sebastien/web/laravel5/vendor/laravel/framework/src/Illuminate/View/ViewFinder.php

```
127.     protected function findInPaths($name, $paths)
128.     {
129.         foreach ((array) $paths as $path) {
130.             foreach ($this->getPossibleViewFiles($name) as $file) {
131.                 if ($this->files->exists($viewPath = $path.'/'.$file)) {
132.                     return $viewPath;
133.                 }
134.             }
135.         }
136.
137.         throw new InvalidArgumentException("View [{$name}] not found.");
138.     }
139.
140.     /**
141.      * Get an array of possible view files.
142.      *
143.      * @param string $name
144.      * @return array
145.      */
146.     protected function getPossibleViewFiles($name)
147.     {
148.         return array_map(function ($extension) use ($name) {
149.             return str_replace('.', '/', $name).'.'.$extension;
```

Arguments

1. "View [articles.create] not found."

No comments for this stack frame.

CREATION D'UN FORMULAIRE : LA VUE

- ▶ On va faire une vue très simple qui contient que le formulaire
- ▶ Ici rien vraiment de nouveau

```
<h1>Création d'un article</h1>
<form method="POST" action="/articles/create">
    {{ csrf_field() }}
    <input name="title" type="text" placeholder="Titre"/>
    <textarea name="body" placeholder="Contenu"></textarea>
    <button type="submit">Enregistrer</button>
</form>
```


CREATION D'UN FORMULAIRE : LA MÉTHODE STORE

- ▶ Cette méthode doit :
 - ▶ Valider les données
 - ▶ Rediriger vers le formulaire si erreur un message
 - ▶ Enregistrer en base de données si les données sont valides avec un message

CREATION D'UN FORMULAIRE : LA VALIDATION

- ▶ Valider les données: on utilise la méthode `validate`
- ▶ On passe les règles à la méthode `validate`.
- ▶ Les règles peuvent être combinées avec `|`
- ▶ Il en existe pleins -> voir la doc !
- ▶ Exemple : `'email' => 'required|email|unique:users'`

Pour notre formulaire

```
public function store()
{
    request()->validate([
        'title' => 'required|max:100',
        'body' => 'required'
    ]);
}
```

CREATION D'UN FORMULAIRE : LA VALIDATION

- ▶ On va récupérer que les données qui auront été épurées et validées
- ▶ Si c'est bon on ajoute l'article (Il faut le modèle et la migration
php artisan make:model Article
etc.)
- ▶ Le si est implicite !

```
public function store()
{
    $data = request()->validate([
        'title' => 'required|max:100',
        'body' => 'required'
    ]);

    Article::create($data);
}
```

CREATION D'UN FORMULAIRE : AFFICHER LES ERREURS

- ▶ Si la validation échoue : le texte des erreurs est envoyé à la session.
- ▶ On fera donc apparaitre le message d'erreur et on garde les bonnes valeurs
- ▶ Tout est dans la variable \$errors

CREATION D'UN FORMULAIRE : AFFICHER LES ERREURS

- ▶ Laravel fournit cette variable `$errors` à toutes les vues.
- ▶ Cette variable possède des méthodes:
 - ▶ Exemple : `$errors->any()` permet de savoir si il y a des erreurs
 - ▶ La méthode `$errors->all()` récupère l'ensemble des erreurs et on pourra itérer dessus
 - ▶ `$errors->get('champ')` : les erreurs de ce champ

CREATION D'UN FORMULAIRE : AFFICHER LES ERREURS

```
<h1>Création d'un article</h1>
@if ($errors->any())
    <strong>
        Il y a des erreurs dans le formulaire
    </strong>
    <ul>
        @foreach ($errors->all() as $error)
            <li>{{ $error }}
        @endforeach
    </ul>
@endif
<form method="POST" action="/articles/create">
    {{ csrf_field() }}
    <input name="title" type="text" placeholder="Titres"/>
    <textarea name="body" placeholder="Contenu"></textarea>
    <button type="submit">Enregistrer</button>
</form>
```

Création d'un article

Il y a des erreurs dans le formulaire

- The body field is required.

Titres	Contenu	Enregistrer
--------	---------	-------------

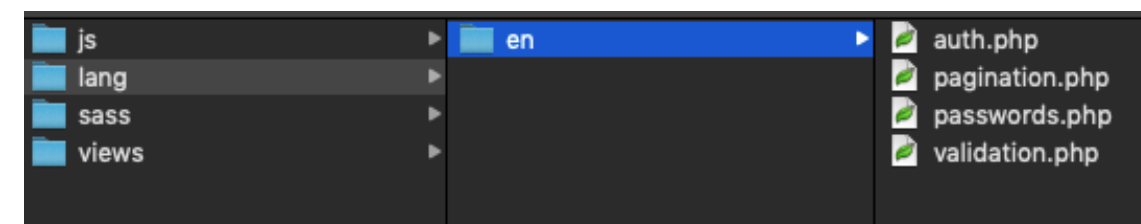
Les erreurs sont affichées en anglais. Pour l'avoir en français il faut aller dans resources/lang

La partie validation est géré par **validation.php**

Il faut faire un copier-coller en le dossier fr et changer le texte.

Sinon certain l'ont déjà fait ^^ A vous de chercher !

Il faut également déclarer le français dans config/app.php



CREATION D'UN FORMULAIRE : RÉCUPÉRER LES ANCIENNES VALEURS

- ▶ Pour éviter la frustration de l'utilisateur il est intéressant de récupérer les anciennes valeurs
- ▶ Laravel met à votre disposition une fonction `old`
- ▶ Simple et efficace
- ▶ Les anciennes valeurs sont passées à la session

CREATION D'UN FORMULAIRE : RÉCUPÉRER LES ANCIENNES VALEURS

```
<form method="POST" action="/articles/create">
    {{ csrf_field() }}
    <input name="title"
    type="text"
    placeholder="Titre"
    value="{{ old('title') }}" />

    <textarea name="body"
    placeholder="Contenu">{{ old('body') }}</textarea>
    <button type="submit">Enregistrer</button>
</form>
```

Création d'un article

Il y a des erreurs dans le formulaire

- The body field is required.

SuperTitreAssezLong	Contenu	Enregistrer
---------------------	---------	-------------

CREATION D'UN FORMULAIRE : VALIDER ET ENVOYER UN MESSAGE

- ▶ C'est la dernière étape : si tout c'est bien déroulé
- ▶ On doit rediriger vers une page. Par exemple ici une page qui listerait des derniers articles ajoutés et avec un message

```
public function store()
{
    $data = request()->validate([
        'title' => 'required|max:100',
        'body' => 'required'
    ]);

    Article::create($data);

    return redirect('articles')->with([
        'message' => 'Votre article a été enregistré']);
}
```

CREATION D'UN FORMULAIRE : VALIDER ET ENVOYER UN MESSAGE

- ▶ Le with passe la valeur à la session
- ▶ Dans la vue articles on testera si il y a une valeur avec `@if (session('message'))`

LE CONTRÔLEUR

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Article;

class ArticleController extends Controller
{

    public function create()
    {
        return view('articles.create');
    }

    public function store()
    {
        $data = request()->validate([
            'title' => 'required|max:100',
            'body' => 'required'
        ]);

        //Article::create($data);

        return redirect('articles')->with([
            'message' => 'Votre article a été enregistré']);
    }

    public function show($n)
    {
        return view('article')->with('numero', $n);
    }
}
```

LA VUE

```
<h1>Création d'un article</h1>
@if ($errors->any())
    <strong>
        Il y a des erreurs dans le formulaire
    </strong>
    <ul>
        @foreach ($errors->all() as $error)
            <li>{{ $error }}
        @endforeach
    </ul>
@endif
<form method="POST" action="/articles/create">
    {{ csrf_field() }}
    <input name="title" type="text" placeholder="Titres"/>
    <textarea name="body" placeholder="Contenu"></textarea>
    <button type="submit">Enregistrer</button>
</form>
```

TAF

- ▶ A partir des connaissances que vous venez de voir. Je vous demande d'établir un formulaire. Il doit respecter toutes les étapes.
- ▶ Au lieu de l'ajouter dans une base il doit envoyer un mail (a vous de configurer Laravel -> voir MailTrap)

Contactez-moi

Votre message

Envoyer !

Contactez-moi

Merci. Votre message à été transmis. Vous recevrez rapidement une réponse.

Accueil / Démo / Contact

2019-03-30 09:38
(Il y a 14 minutes)
Taille: 759 octets

Contact

De: Sébastien Inion <contact@inion.info>
Pour: <contact@inion.info>
[Plus d'informations](#)

HTML

Source HTML

Texte

Brut

Une analyse

Vérifier le HTML

Prise de contact sur mon beau site

Réception d'une prise de contact avec les éléments suivants:

- **Nom** : Sébastien
- **Email** : contact@inion.info
- **Message** : test mail depuis Laravel

TAF

- ▶ Voici un exemple de template

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Mon joli site</title>
    {!! Html::style('https://netdna.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css') !!}
    {!! Html::style('https://netdna.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-theme.min.css') !!}
    <!--[if lt IE 9]>
      {{ Html::style('https://oss.maxcdn.com/libs/html5shiv/3.7.2/html5shiv.js') }}
      {{ Html::style('https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js') }}
    <![endif]-->
    <style> textarea { resize: none; } </style>
  </head>
  <body>
    @yield('contenu')
  </body>
</html>
```

- ▶ Votre formulaire doit compléter -contenu-
- ▶ Le nom est obligatoire, compris entre 5 et 20 caractères
- ▶ Le mail est requis et ça doit être une adresse mail
- ▶ Le texte est requis et doit comporter maximum 250 caractères
- ▶ On envoie le mail que si il n'y a pas d'erreur