Protection CSRF(Cross-Site Request Forgery)

Falsification de requêtes intersites

Les falsifications de requêtes inter-sites sont un type d'exploit malveillant par lequel des commandes non autorisées sont exécutées au nom d'un utilisateur authentifié. L'impact de cette attaque dépend des privilèges de la victime sur le système.

Laravel fournit une protection contre les attaques CSRF en générant un jeton CSRF . Ce jeton CSRF est généré automatiquement pour chaque utilisateur. Ce jeton n'est rien d'autre qu'une chaîne aléatoire gérée par l'application Laravel pour vérifier les demandes des utilisateurs.

Cette protection de jeton CSRF peut être appliquée à n'importe quel formulaire HTML dans l'application Laravel en spécifiant un champ de formulaire caché du jeton CSRF.

Une des manières efficaces de se prémunir contre les attaques CSRF est d'accompagner chaque formulaire à protéger d'un jeton (en anglais "token") d'identification de session. Pour ce faire :

- ➔ lorsqu'un visiteur se connecte sur votre site, vous enregistrez en session une chaîne de caractère aléatoire.
- ➔ Cette même chaîne est récupérée en session et passée au formulaire, sous la forme d'un champ de type "hidden".
- ➔ Lorsque le formulaire est posté, votre système de validation doit vérifier que le champ caché contient la même valeur que celle enregistrée en session.
- ➔ Si ça n'est pas le cas, ça signifie que quelqu'un a essayé de soumettre le formulaire depuis un autre endroit que votre site. Il faut alors annuler le traitement de ce formulaire.

Laravel inclut un plug-in CSRF intégré, qui génère des jetons pour chaque session utilisateur active. Ces jetons vérifient que les opérations ou requêtes sont envoyées par l'utilisateur authentifié concerné.

Empêcher les requêtes CSRF

CSRF est implémenté dans les formulaires HTML déclarés dans les applications Web. Vous devez inclure un jeton CSRF validé masqué dans le formulaire, afin que le middleware de protection CSRF de Laravel puisse valider la demande.

Cours Proposé par : ELGARRAI Zineb

La protection CSRF peut être implémentée dans Laravel à l'aide de n'importe quel formulaire HTML avec une forme cachée de jeton CSRF et la demande de l'utilisateur est validée à l'aide du middleware CSRF VerifyCsrfToken.

L'une des options suivantes peut être utilisée pour générer un jeton CSRF :

1- Empêcher les requêtes CSRF via @csrf

@csrf est une directive de Blade pour générer un champ de jeton qui sera utilisé pour la vérification. Il génère un champ de saisie masqué.

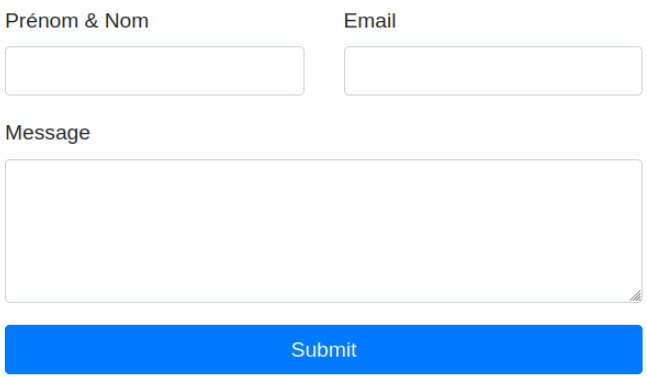
Lorsque la directive blade @csrf est utilisée, une balise **input** de type='**hidden**' est ajoutée au formulaire. La valeur sera le jeton CSRF qui sera envoyé dans le cadre des données du formulaire lorsque le formulaire est soumis.

Syntaxe :

```
<form action="{{ url('/.....') }}" method="post">
    @csrf
    ...
</form>
```

Exercice

1- Créer une vue qui contient un formulaire comme suit :



dans l'attribut action on a : /accueil (penser à créer une page accueil)

2- Tester le bon fonctionnement sans et avec l'utilisation du : @csrf

2- Empêcher les requêtes CSRF via csrf_token ()

csrf_token () : Cette fonction peut être utilisée dans la balise meta et le champ de saisie masqué du formulaire HTML. Il génère une chaîne aléatoire en tant que jeton CSRF.

Syntaxe :

```
<form method = "POST">
    <input type = "hidden" name = "_ token" value = "{{csrf_token ()}}">
        .....
        .....
</form>
```

Exercice :

Créez une vue Laravel nommé *form-csrf_token.blade.php* où la fonction **csrf_token()** est utilisée pour générer le jeton CSRF.

3- Empêcher les requêtes CSRF via csrf_field()

csrf_field () : Cette fonction crée un champ masqué pour le formulaire HTML où il est utilisé et génère un jeton CSRF.

Syntaxe :
```
<form method = "POST“ action="/profile">
    {{ csrf_field() }}
     .....
</form>
```

Exercice :

Créez une vue Laravel nommé *form-csrf_field.blade.php* où la fonction **csrf_token ()** est utilisée pour générer le jeton CSRF.

Remarque : Exclure les URI de la protection CSRF

Laravel a la protection CSRF activé par défaut pour toutes les demandes qui transitent par votre application. Parfois, vous souhaiterez peut-être exclure un ensemble d'URI de la protection CSRF (surtout lorsqu'on utilise des services extrenes dont on n'a pas droit de définir un jeton).

Aisni, on peut exclure les routes en ajoutant leurs URI à la propriété *$except* du *middleware VerifyCsrfToken*

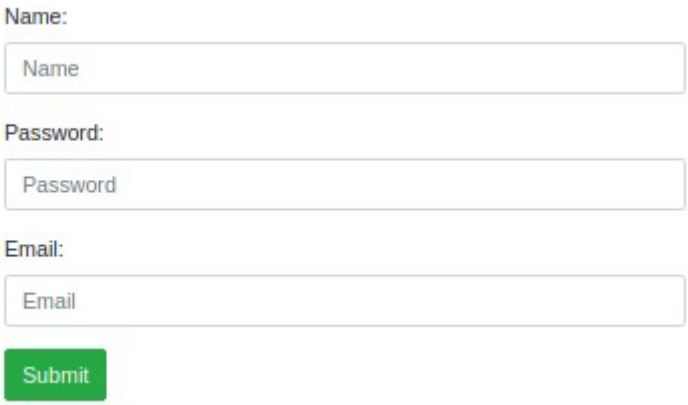
```
....
class VerifyCsrfToken extends Middleware
{
        protected $except = [

                'les routes  à exclues'
        ];
}
```

Récupérer les paramètres d'un formulaire

On traite cette partie sous forme d'exemple à suivre :

1- Créer une vue qui contient le formulaire suivant :

NB : Les éléments à envoyer doivent avoir un nom (l'attribut name)



La partie @csrf est très important, elle permet de s'assurer que les informations parviennent d'un formulaire du site Laravel et évite le piratage. La partie @csrf est remplacer par un code généré automatiquement par Laravel. Ce code qui est retourné par les formulaire à chaque envoie des données.

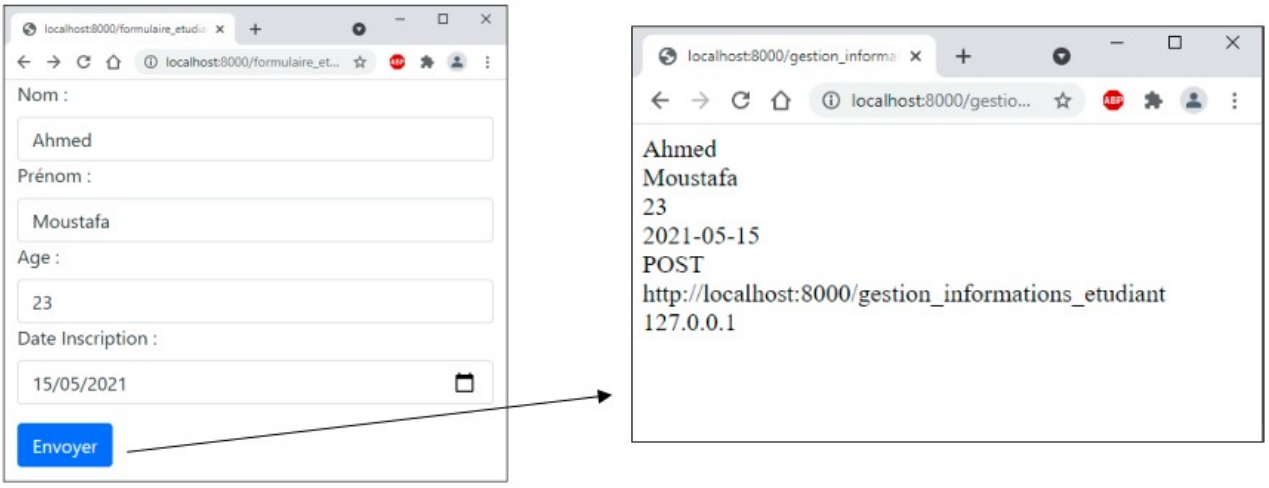
2- Créer un nouveau contrôleur qui a une méthode getData qui permet de récupérer les informations de l'étudiant à partir de son attribut $request, puis retourne une vue 'ReadData' dont elle envoi les données récupérées.

```
public function getData(Request $request){
$data=[];
$data['name']=$request->input('name');
$data['email']=$request->input('email');
$data['pw']=$request->input('password');
return view('readForm',$data);
}
```

3- Créer la vue 'ReadData.blade.php' qui permet d'afficher les données comme suit :

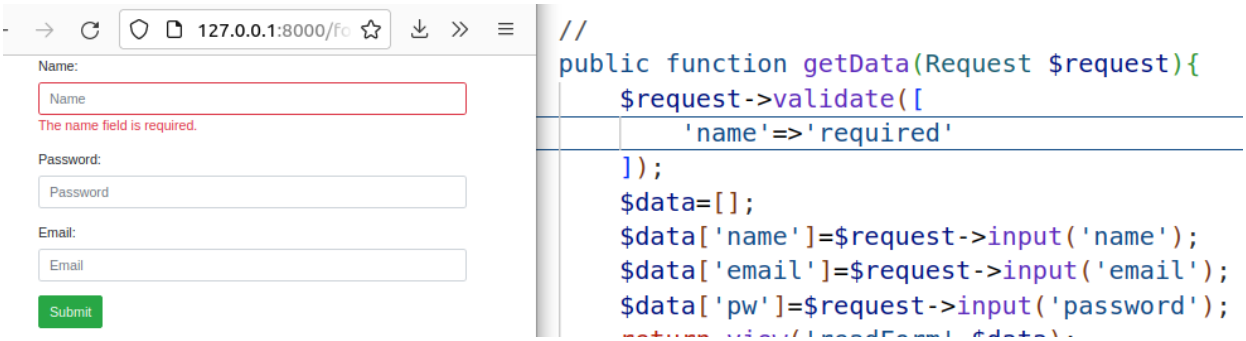
Exercice :

Suivez les mêmes étapes pour ce formulaire :

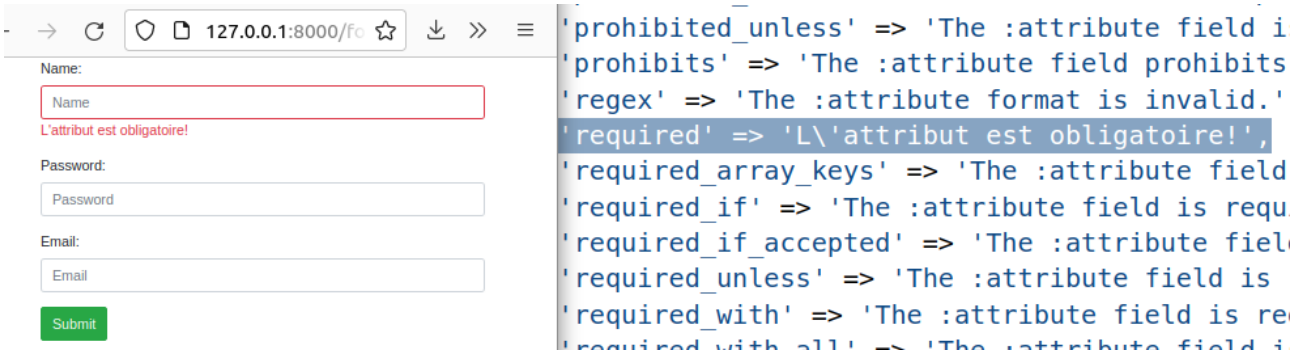


La validation de formulaires

La validation de formulaire permet de vérifier la cohérence les données d'un formulaire avant de les sauvegarder dans la base de données et cela en définissant un ensemble de règle à respecter. On va prendre comme exemple l'ajout d'un nouveau étudiant dans la base de données. Avant d'ajouter un nouveau étudiant on doit vérifier si les données du formulaire sont valides comme suit :



Les messages d'erreur sont en anglais. On peut cependant personnaliser les messages en modifiant le fichier : « \lang\validation.php »

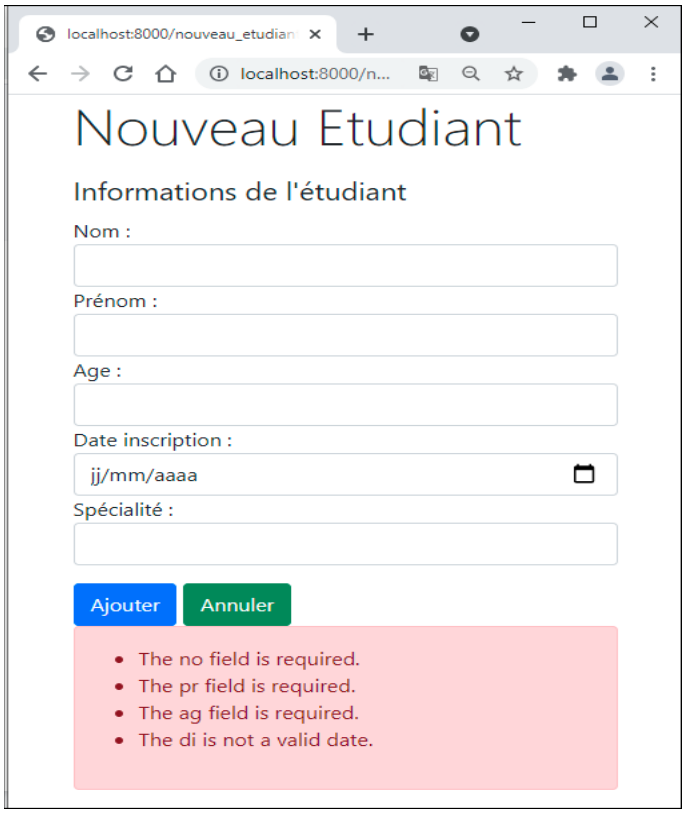


Pour afficher les messages d'erreurs on modifie la vue du formulaire « readForm.blade.php » en ajoutant le morceau de code suivant après chaque zone de texte :

```
@error('name')
<span class="text-danger">{{ $message }}</span>
@enderror
```

Exercice :

Compléter l'exercice précédent pour pouvoir lister les erreurs suivantes :



indications :

1- Utiliser une boucle foreach sur la liste des erreurs ($errors→all())

2- vous pouvez verifier si on a des erreurs avec if ($errors →any()) ...