UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 2 Informatique
INFO0303 – 2019/2020
***Resp. :* F. Alin**

Travaux Pratiques n° 6

Utilisation d'Ajax avec Laravel

1 Utilisation d'Ajax

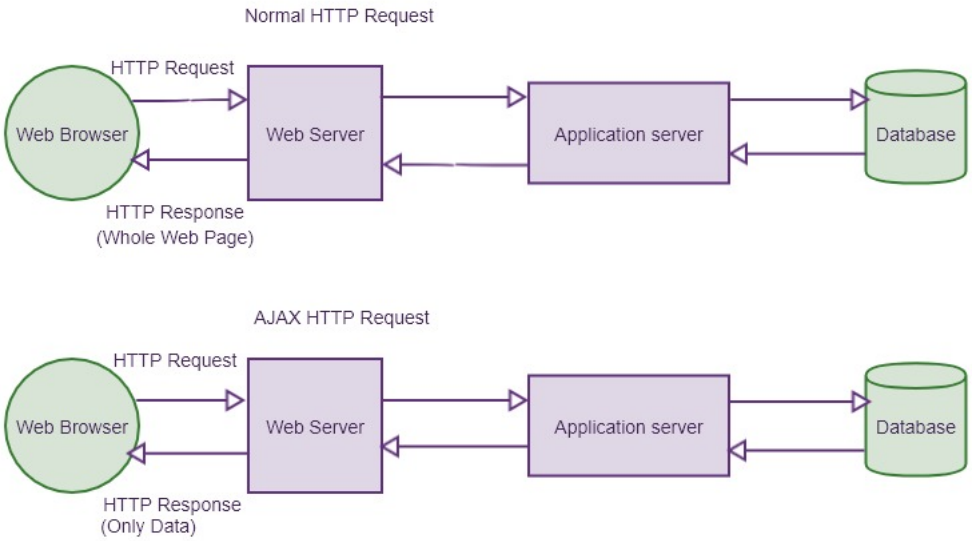
1.1 Introduction

AJAX permet de communiquer entre le client et le serveur sans actualiser la page. Cette technique permet de transmettre des données d'un serveur à un autre sans interruption. AJAX signifie **Asynchronous Javascript and XML** Quand on utilise un formulaire standard, on définit une route POST dans la propriété action. Dans le cas d'AJAX, on délègue ce travail à un code javascript qui gère côté client l'ensemble de l'échange.

1.2 Différence entre une requête normale et une requête Ajax

Remarque :

Dans le cas d'une requête "normale", le client transmet une demande au serveur, qui répond en fournissant la page demandée, page qui contient à la fois des données statiques et des données dynamiques.
Lors d'une demande AJAX, le serveur répond uniquement en retournant les données et non une page entière.
Le serveur renvoie généralement les données au format JSON (ou au format XML)



1.3 Mise en œuvre

Nous allons utiliser AJAX dans Laravel. Tout d'abord, nous allons créer un formulaire simple à l'aide de bootstrap. Ensuite, en utilisant jQuery, nous enverrons une demande au serveur pour stocker les valeurs de formulaire dans la base de données. Commençons donc cette petite application en installant le Laravel.

Etape 1 : Installez et configurez le Laravel.

Installez une instance vierge de Laravel :

```
composer create-project laravel/laravel laravel-ajax --prefer-dist
```

Configurez la base de données, en complétant les données d'authentification de base de données dans le fichier .env.

Nous allons construire une application de gestion de news, dans laquelle l'utilisateur peut ajouter des articles. Commençons pour cela par construire un modèle. Générez un fichier de migration, à l'aide de la commande suivante :

```
php artisan make:migration create_news_table
```

Complétez le fichier de migration, afin de définir les colonnes de votre table :

```
 public function up()
    {
        Schema::create('news', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->string('auteur');
            $table->string('message');
            $table->string('date');
            $table->timestamps();
        });
    }
```

Puis réalisez la migration de la table :

```
php artisan migrate
```

Étape 2 : Définissez les routes, le modèle et les contrôleurs.

Générez un modèle et un contrôleur, avec les commandes suivantes :

```
php artisan make:model News
php artisan make:controller NewsController
```

L'étape suivante consiste à définir les routes en complétant le fichier web.php :

```
// web.php

Route::view('/news', 'news');
Route::post('/news/post', 'NewsController@store');
```

Créez maintenant une vue en ajoutant le fichier `news.blade.php` dans le répertoire ressources/vues :

```
<!doctype html>
<html lang="fr">
<head>
```

```
    <title>Lecteur de News</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/
</head>
<body>
    Lecteur de News
</body>
</html>
```

Étape 3 : Créez un formulaire Bootstrap.

Complétez la vue news.blade.php, avec le code suivant :

```
<body>
    <div class="container">
        <form id="myForm">
            <div class="form-group">
                <label for="nom">Nom de l'auteur :</label>
                <input type="text" class="form-control" id="nom"> </div>
            <div class="form-group">
                <label for="message">Nouvelle :</label>
                <input type="text" class="form-control" id="message">
            </div>
            <div class="form-group">
                <label for="date">Date:</label>
                <input type="text" class="form-control" id="date">
            </div>
            <button class="btn btn-primary">Envoyer</button>
        </form>
    </div>
</body>
```

Afin de simplifier la mise en place des échanges entre le client et le serveur, nous allons utiliser AJAX. Pour cela nous allons importer la librairie jQuery dans le fichier news.blade.php. Dans l'exemple suivant nous utilisons la version CDN de jQuery, mais vous pouvez aussi utiliser une version locale du fichier jQuery.

```
<script src="http://code.jquery.com/jquery-3.3.1.min.js"
                integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
                crossorigin="anonymous">
</script>
```

Étape 4 : Configurez une requête Ajax pour Laravel.

Pour tout échange de données, Laravel exige l'utilisation du jeton CSRF habituellement défini dans une balise méta en insérant un champ `csrf_field()`.

```
<meta name="csrf-token" content="{{ csrf_token() }}">
```

Avec Ajax, lorsque l'on configure une requête, il faut en plus configurer l'en-tête de la requête afin qu'elle contienne le jeton CSRF. L'intégration du token est réalisée par le code suivant :

```
$(document).ready(function(){
    $('#ajaxSubmit').click(function(e){
```

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

Lorsque l'utilisateur clique sur le bouton d'envoi, l'événement click est appelé. On empêche d'abord l'envoi du formulaire, puis on définit l'en-tête de demande de d'enregistrement.

On complète le script avec un nouvel appel à la fonction `jQuery.ajax ()` pour soumettre la demande au serveur avec les trois paramètres d'entrée.

```
......
 $.ajax({
     url: "{{ url('/news/post') }}",
     method: 'post',
     data: {
         nom: $('#nom').val(),
         message: $('#message').val(),
         date: $('#date').val()
     },
     success: function(result){
         $('.alert ').show();
         $('.alert ').html(result.success);
     }});
```

La fonction `$.ajax ()` reçoit un objet en paramètres. Celui-ci contient l'URL pour envoyer la requête, la propriété `method` pour définir quelle méthode de requête HTTP est utilisée, un objet de données `data` contenant toutes les données du formulaire. Il est également possible de définir des fonctions `success` et `error` qui permettent de définir les traitements à effectuer suivant que la requête s'est correctement déroulée, ou au contraire a abouti à une erreur de traitement.

Étape 5 : écrivez la fonction de stockage pour stocker les données.

Ajoutez dans le fichier NewsController.php le code suivant :

```
use App\News;

class NewsController extends Controller
{
    public function store(Request $request) {
        $news = new News();
        $news->auteur = $request->nom;
        $news->message = $request->message;
        $news->date = $request->date;
        $news->save();
        return response()->json([
            'success'=>'Vos données ont été correctement enregistrées'
        ]);

    }
}
```

Il ne reste plus qu'à modifier la vue afin d'afficher un message indiquant que les données ont été enregistrées avec succès.

```
<!doctype html>
<html lang="fr">
<head>
    <title>Lecteur de News</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">

</head>
<body>
    <div class="container">
        <form id="myForm">
            <div class="alert alert-success" style="display:none"></div>
            <div class="form-group">
                <label for="nom">Nom de l'auteur :</label>
                <input type="text" class="form-control" id="nom"> </div>
            <div class="form-group">
                <label for="message">Nouvelle :</label>
                <input type="text" class="form-control" id="message">
            </div>
            <div class="form-group">
                <label for="date">Date:</label>
                <input type="text" class="form-control" id="date">
            </div>
            <button class="btn btn-primary" id="ajaxSubmit">Envoyer</button>
        </form>
    </div>

    <script src="http://code.jquery.com/jquery-3.3.1.min.js"
            integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
            crossorigin="anonymous">
    </script>
    <script>
        $(document).ready(function(){
            $('#ajaxSubmit').click(function(e){
                e.preventDefault();
                $.ajaxSetup({
                    headers: {
                        'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
                    }
                });
                $.ajax({
                    url: "{{ url('/news/post') }}",
                    method: 'post',
                    data: {
                        nom: $('#nom').val(),
                        message: $('#message').val(),
                        date: $('#date').val()
                    },
                    success: function(result){
                        $('.alert ').show();
                        $('.alert ').html(result.success);
```

```
                    });
            });
        });
    </script>
</body>
</html>
```

2 Exercice

1. mettre en œuvre l'exemple ci-dessus ;
2. modifiez sur la vue `news` afin qu'elle affiche les 5 derniers messages entrés ;
3. modifiez cette page afin qu'il soit possible de modifier/supprimer l'un des messages affichés.