

## Workshop : Ajax et JQuery

### Objectif

L'objectif de ce workshop est de créer un système de manipulation de formulaires Ajax simple, dynamique et sans chargement de page. En effet, ce système fournit le moyen de charger dynamiquement du contenu, des scripts et des données et de les utiliser sans avoir à recharger la page et attendre que le serveur ait fini de répondre à la requête.

### I- Définition et principe de fonctionnement d'AJAX

#### 1. Définition

- Ajax permet d'interroger le serveur sans recharger intégralement la page.
- La gestion des événements se fait de façon asynchrone.
- Le format de données JSON (JavaScript Object Notation) constitue le standard actuel pour les échanges de données sur le Web, notamment avec AJAX. Il s'agit d'une syntaxe pour décrire des informations structurées :

```
{ "modèle" : "Peugeot",  
  "couleur" : "bleu",  
  "immatriculation" : 2008,
```

- Ce format permet d'échanger des objets entre le client et le serveur.

#### 2. Principe de fonctionnement

1. Le navigateur envoie une requête HTTP au serveur :
  - ✓ On peut utiliser la méthode GET ou la méthode POST.
  - ✓ Les informations envoyées sont encodées au format JSON.
2. Le serveur traite la demande (html, php, texte, ...)
3. La réponse est récupérée par le navigateur, qui appelle une fonction JavaScript pour traiter la réponse.
4. La réponse du serveur peut contenir du HTML, du texte, format JSON.

#### 3. Ajax et JQuery

- La bibliothèque jQuery ajoute des avantages pour Ajax.
- Pour les communications asynchrones avec le serveur, jQuery offre plusieurs méthodes pour extraire des données Ajax (XML, JSON, script, HTML ou text)
- La méthode jquery \$.ajax() ne crée pas un objet jQuery mais une requête XMLHttpRequest. On ne manipule plus le DOM, on gère une interface asynchrone avec le serveur Web

- Elle fournit le moyen de charger dynamiquement du contenu
- Cette méthode renvoie une référence sur l'objet XMLHttpRequest qui est créé lors de son exécution
- Le principe de fonctionnement de l'objet XMLHttpRequest est d'envoyer une requête HTTP vers le serveur, et une fois la requête envoyée, les données renvoyées par le serveur peuvent être récupérées.

#### 4. La méthode \$.ajax

La méthode \$.ajax est spécifiée génériquement pour tous les objets de la collection référencée par \$

Exple

```
$.ajax({
  type: "POST",
  url: "test.php",
  success: function(retour){
    alert("Données retournées : " + retour);
  }
});
```

Paramètre :  
un objet au format JSON  
(equiv. tableau associatif)

Les sorties du script PHP invoqué,  
test.php, sont accessibles  
dans le paramètre *retour*

👉 **type**

GET ou POST

👉 **url**

fichier désiré sur le serveur WEB (html, php, ...)

👉 **success**

(référence) fonction invoquée quand la requête a  
abouti positivement.  
Le paramètre est le texte généré par l'URL.

#### Cas d'une requête sur un lien :

Ici, AJAX est lancé  
au clic sur le lien  
*En laissant href="..." dans le lien,  
cela permet un fct même si JS est  
bloqué par le navigateur.*

Exple avec un lien :

```
<a href="script.php"
  class="test">
</a>
<div id="rpse">
</div>
```

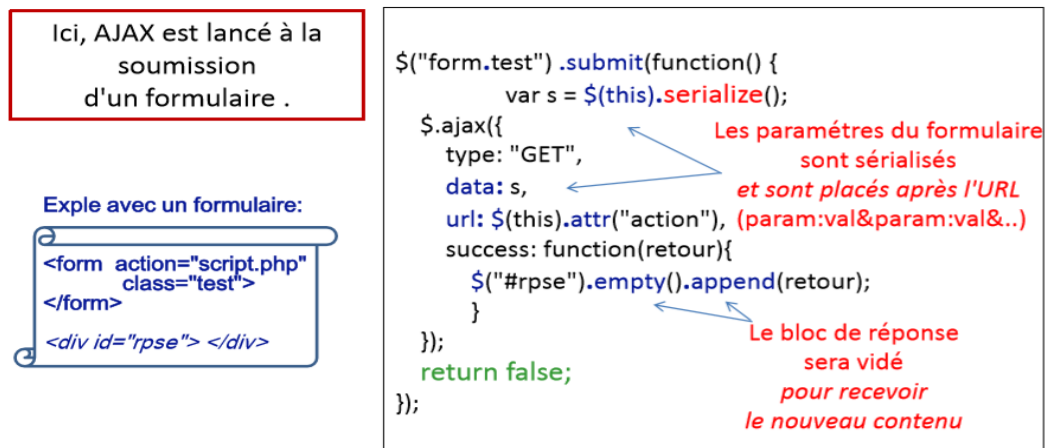
Exple

```
$("a.test").click(function() {
  $.ajax({
    type: "POST",
    url: $(this).attr("href"),
    success: function(retour){
      $("#rpse").empty().append(retour);
    }
  });
  return false;
});
```

La valeur de l'attribut *href*  
Récupérée est celle  
du lien courant, cliqué.

élimine toute propagation  
aux restes des gérants  
événementiels du lien

#### Requête en méthode GET sur un formulaire



### Paramètre de \$.ajax précisant les formats d'échange

- **contentType** : Type de contenu envoyé (par défaut : application/x-www-form-urlencoded).
- **dataType** Type de contenu reçu (par défaut : html)
  - ✓ **xml**: texte xml
  - ✓ **html**: texte sans mise en forme mais les scripts sont interprétés.
  - ✓ **script**: texte JS sans mise en forme.
  - ✓ **json**: texte JS au format JSON, qui sera directement interprété -> retour d'un objet Javascript.

### Les raccourcis \$.get() et \$.post()

- \$.get(url,données,callback,typeResultat)
  - ✓ **url**: chaîne de caractères contenant l'adresse où la requête est envoyée.
  - ✓ **data**: objet JSON, peut aussi être une chaîne de caractères.
  - ✓ **callback**: fonction appelée lorsque le navigateur reçoit le résultat.
  - ✓ **typeResultat(optionnel)**: chaîne de caractères permettant de demander au serveur le type de données du résultat (xml, json, script, text, html).
- \$.post(url, données, callback, typeResultat).

## II- Prérequis

Soit une liste des étudiants récupérée d'une base de données qui contient une entité

« Student » dont les attributs sont:

- ✓ **id** : clé primaire, integer
- ✓ **nsc** : string
- ✓ **email** : string

L'objectif de cette partie est de réaliser une recherche instantanée sans avoir recharger la page et attendre que le serveur ait fini de répondre à la requête (Figure1).

RF			
ID	NSC	Email	Editer
	0596AQ02	ons.ali@esprit.tn	modifier supprimer
	0999RF02	amal.jmal@esprit.tn	modifier supprimer
	7777KL02	unit.esprit@esprit.tn	modifier supprimer
	0596AQ88	uu@tp	modifier supprimer
	059996po	uu@ttpo	modifier supprimer



ID	NSC	Email	Editer
	0999RF02	amal.jmal@esprit.tn	modifier supprimer

On ajoute un champ texte pour faire la recherche dont l'id est « search » :

`<input type="text" id="search" class="form-control" placeholder="Search">`

### III- Recherche instantanée en Ajax avec Symfony 4

1. Pour récupérer de la base de données la liste des étudiants désirés, il faut, tout d'abord, créer une fonction nommée “findStudentByNsc” dans le repository Student en ajoutant le code suivant :

```
public function findStudentByNsc($nsc){
    return $this->createQueryBuilder('student')
        ->where('student.nsc LIKE :nsc')
        ->setParameter('nsc', '%'.$nsc.'%')
        ->getQuery()
        ->getResult();
}
```

2. Afin de faire la recherche et obtenir la liste des étudiants désirés, il faut créer une fonction nommée “searchStudentx” dans votre contrôleur en ajoutant le code suivant :

```
/**
 * @Route("/searchStudentx ", name="searchStudentx")
 */
public function searchStudentx(Request $request, NormalizerInterface $Normalizer)
{
    $repository = $this->getDoctrine()->getRepository(Student::class);
    $requestString=$request->get('searchValue');
    $students = $repository->findStudentByNsc($requestString);
    $jsonContent = $Normalizer-
>normalize($students, 'json', ['groups'=>'students']);
    $retour=json_encode($jsonContent);
    return new Response($retour);
}
```

3. Afin d’afficher la liste des étudiants dans un tableau et d’effectuer la recherche instantanée, il faut ajouter le code twig suivant dans le fichier « student.html.twig » :

```

{% extends 'base.html.twig' %}
{% block body %}
/* Cette partie correspond à l’affichage de la liste des étudiants en
listant leurs informations dans un tableau */

<h1 id="dd1"> Liste des étudiants</h1>
</br>
<div style="margin-right:50px;" class="btn btn-primary" ><a href="{{
path('addStudent') }}"> Ajouter Etudiant</a></div>
    </div>
        <input type="text" id="search" class="form-control"
placeholder="Search">
<div>
<table border ="1" id="t" class="table table-hover table-dark">
    <thead class="thead-dark">
        <tr>
            <td >ID</td>
            <td >NSC</td>
            <td >Email</td>
            <td >Editer</td>
        </tr>
    </thead>
    <tbody id="all">
        {% for student in students %}
    <tr>
        <td>
{{ student.id }}
        </td>
        <td>
{{ student.nsc }}
        </td>
        <td>
{{ student.email }}
        </td>
        <td>
<a href="{{ path('updateStudent', {'id': student.id}) }}"> modifier</a>
<a href="{{ path('deleteStudent', {'id': student.id}) }}"> supprimer</a>
        </td>
    </tr>
        {% endfor %}
    </tbody>
    <tbody id="search">
    </tbody>
</table>
</div>
    {% endblock %}
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
<script>
$( document ).ready(function() {
    $("#search").keyup(function(e){
/* La variable value va prendre la valeur insérer dans le champ de texte
afin d’effectuer la recherche */
        var value = $(this).val();
/* Ajax est lancé lors du remplissage du champ texte dont l’id est
« search » pour faire la recherche */

```

```

$.ajax({
/* l'url est une chaine de caractères contenant l'adresse où la requête est
envoyée */
    url : "{{ path('studentListx') }}",
/* La méthode utilisée pour transférer les données est GET */
    type : 'GET',
/*Ici search value va prendre la chaine entrée par un utilisateur dans la
zone de recherche et sera placée après l'url */
    data: {
        'searchValue' : value
    },
/*Cette fonction permet de vider le contenu du tableau pour recevoir le
nouveau contenu*/
    success : function(retour){
        if(retour){
            $('#tbody#search').empty();
            $.each(JSON.parse(retour), function(i, obj) {
                $('#tbody#all').hide();
                $('#tbody#search').append('<tr><td> '+obj.id+'
</td><td> '+obj.nsc+' </td><td>'+obj.email+' </td><td><a
href="updateStudent/'+obj.id+'">modifier</a> </br><a
href="deleteStudent/'+obj.id+'">supprimer</a></td></tr>');
            });
        }
        else
        {
            $('#tbody#all').show();
            $('#tbody#search').empty();
            $('#tbody#search').fadeIn('fast');
        }
    },
    });
    return false;
    });
});
</script>

```