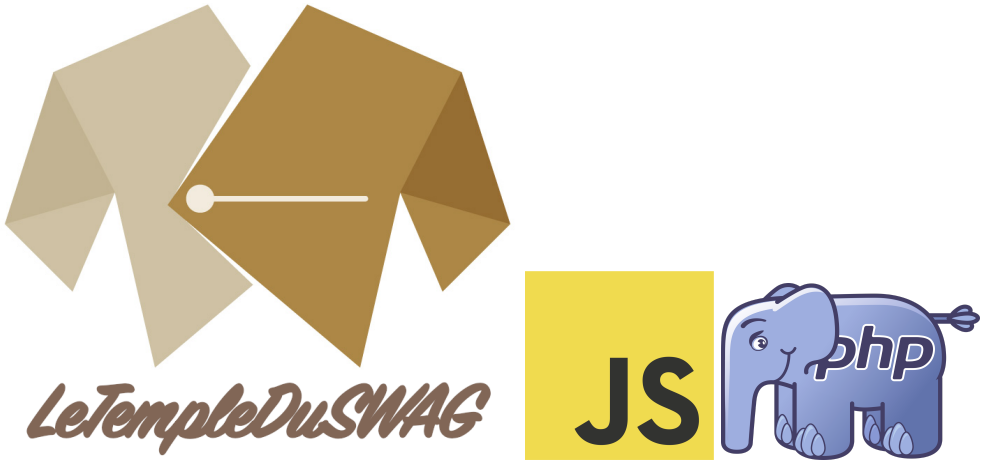
# Web avancé – L2 MIASHS

## TD 10 – LeTempleDuSwag – fonctionnalités d'ajout et suppression




Votre site du temple du Swag permet désormais de filtrer les vêtements par le genre destiné. Cependant, il serait bon d'y ajouter des fonctionnalités d'ajout et de suppression n'est-ce pas ? Dans ce TD, nous allons voir comment ajouter ces deux fonctionnalités tout en conservant une approche asynchrone afin de voir la modification de la page en temps réel. Pour simplifier ce TD, l'ajout sera ici incomplet : il ne considérera pas l'ajout d'image, ni toutes les informations par produit.

Voici notre planning très simplifié :

Feuille de route du TempleDuSwag :

- ☑ ~~Mise en place du parcours des articles~~
- ☑ ~~Ajout d'un filtre par genre en AJAX ()~~
- ☐ Ajout des fonctionnalités de suppression et d'ajout avec formulaire dédié (ce TD 😀)
- ☐ Ajout d'une inscription et d'une connexion de l'utilisateur (TD suivant)

Pour cela nous aurons besoin des éléments suivants :

- un serveur qui fournit des données
- des requêtes asynchrones
- un formulaire pour ajouter un article

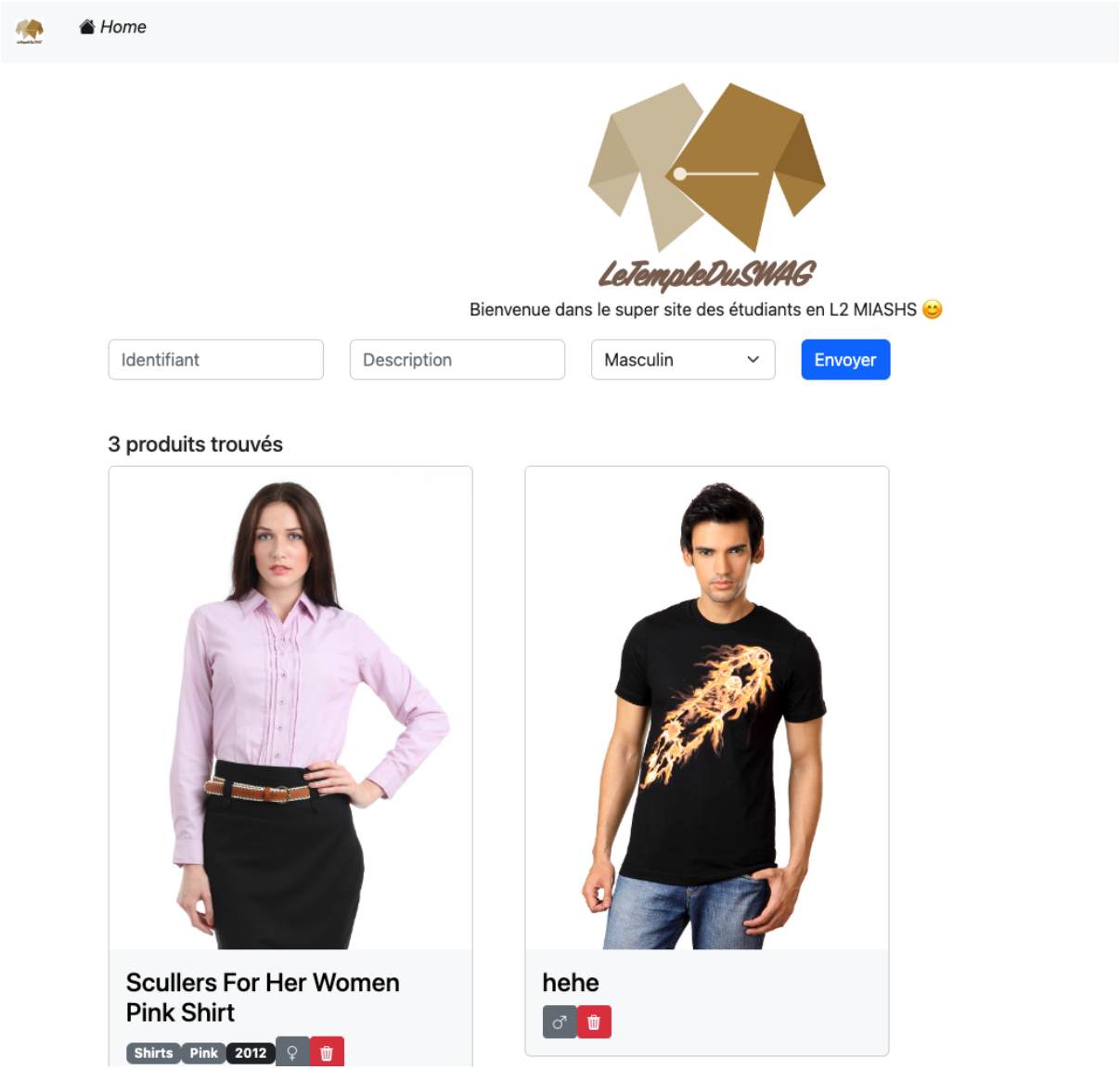
Vous constaterez rapidement que la suppression est bien plus rapide à mettre en place que l'ajout. Ce dernier nécessitant la mise en place d'un formulaire.

**Objectif :** mettre en place la suppression et l'ajout d'articles vestimentaires
**Cible :** les étudiants connaissant déjà Javascript, ayant une bonne connaissance de l'AJAX et à l'aise avec PHP et les requêtes en base de données MySQL
**Durée :** très variable, en fonction de votre aisance dans l'usage combiné des différentes notions.

Un conseil important : même si ce TD peut, à certains points, vous sembler facile, ne sautez pas d'étape !

Chaque exercice se distingue par ce logo  !
Dans ce TD, le code est intégré sous forme d'image afin que vous ne puissiez pas simplement le copier-coller.
Les requêtes SQL vous seront données, il sera à votre charge de les adapter dans votre code PHP. Enfin, certaines explications données dans les TDs précédents ne seront pas systématiquement répétées !

L'objectif du TD est d'atteindre ce résultat :




Dans cette page, le bouton rouge avec l'icône de la poubelle supprimera en un clic l'élément concerné, tandis qu'au-dessus des cartes d'articles est présent un formulaire pour en ajouter.

# Suppression d'un article

Supprimer un article nécessite plusieurs choses :

- un bouton dédié
- un comportement associé à ce bouton sur l'événement de clic
- une gestion de la requête HTTP côté PHP
- une exécution de requête SQL côté PHP

Allons-y dans cet ordre.

## Client – Bouton de suppression

Le bouton de suppression est principalement constitué d'éléments avec des classes bootstrap. Dans les faits vous devez, après le bouton de filtre de genre, suivre les instructions suivantes :

- Créez une balise button (bouton) ayant pour classes "btn" "btn-danger" (pour la couleur du danger, ici rouge 🙂) et "btn-sm" (sm signifiant small – petit en français).
- Ajoutez à ce bouton l'attribut "onclick" qui va permettre de lancer une fonction nommée **remove()** prenant comme argument l'identifiant de l'article. Par exemple: remove(5) pour l'article ayant l'identifiant 5.
- Dans ce bouton, mettez y une icône en tant que contenu HTML comme ceci :

```
<i class="bi bi-trash3-fill"></i>
```

Il nous reste encore à adapter la réécriture du DOM en intégrant ce nouveau bouton. Pour cela, modifiez la fonction **insert()** pour faire, en javascript, ce que vous venez de faire manuellement.

Et voilà pour le bouton ! Son comportement est mis en place mais ne fait rien car la fonction **remove()** n'existe pas encore dans le code javascript. À nous de la créer !

## Client – Comportement du bouton

Il convient de bien prendre en compte le comportement du bouton dans notre fichier **swag.js**. Pour cela :

- Créez une fonction **remove()** acceptant un identifiant comme argument.
- Faites le simplement lancer une fonction dédiée aux modifications des données *E que s'apelerio Q*… euh **modif()** 😅. Cette fonction accepte un seul argument. Donnez lui un tableau associatif contenant l'identifiant et l'action ("delete" pour la suppression, "add" pour l'ajout).

Jusqu'ici tout va bien. Mais il nous créer le principal avec la fonction asynchrone **modif()**.

- Créez une fonction asynchrone nommée **modif()** qui accepte un seul argument.
- Dans cette fonction, initialisez la classe **Headers()** dans une nouvelle variable. Cette classe est dédiée au stockage des headers (les entêtes HTTP).
    - Utilisez ensuite la méthode append() pour y mettre un “Content-Type” de valeur “application/x-www-form-urlencoded; charset=UTF-8”. Cela permettra de contrôler l’encodage. (Pour info nous n'utilisons pas le header “application/json” car son usage est moins simple du côté PHP et demande des étapes supplémentaires.)
- Créez un tableau associatif nommé **config** dans lequel vous allez mettre
    - une clé “method” avec la valeur “POST” (car quand on modifie des données on utilise la méthode POST a minima).
    - une clé “headers” ayant pour valeur la variable contenant les headers
    - une clé “body” contenant le corps de la requête, c’est-à-dire le tableau associatif provenant de l’argument de la fonction mais au format String. Pour convertir un tableau associatif au format String, utilisez **JSON.stringify()**.
- Le reste de la fonction **modif()** consiste à utiliser fetch() / await comme vu précédemment. Attention à simplement bien utiliser la méthode fetch avec l’url de base et la configuration précédemment créée. Par exemple comme ceci : `fetch("http://localhost:8888/td10/api.php", config);`. Il n’y a donc plus besoin de créer une longue URL avec des requêtes en paramètre comme par la méthode GET du précédent TD ! 👍

Oui, vous ne pouvez toujours pas tester la fonctionnalité. Nous avons pour le moment uniquement traité le côté client. Il convient désormais de s’attaquer au serveur.

## Serveur – Contrôle de la requête

Passons à **api.php**. Votre requête est envoyée par le biais de la méthode **fetch()** de Javascript. C’est super, mais il convient désormais de contrôler le comportement du serveur en fonction de la requête. Nous savons que les requêtes de modification vont venir par la méthode POST de HTTP, il nous faut donc la prendre en compte.

- En dessous des conditions précédentes pour la méthode GET, ajoutez une condition vérifiant si la méthode de requête est bien égale à “POST”. Pour ce faire, utilisez la variable **$_SERVER** et sa clé “REQUEST_METHOD”.
- Si la requête utilise la méthode POST, alors ajoutez la ligne suivante `$_POST = json_decode(file_get_contents('php://input'), true);`. L’usage de cette ligne est nécessaire car la variable **$_POST** en PHP ne récupère d’informations uniquement issues d’entêtes “multipart/form-data” ou “application/x-www-form-urlencoded”. Cette ligne permet donc de transformer le contenu issu de la seconde valeur et le transforme en tableau associatif à partir du JSON précédemment mis au format String (**JSON.stringify()** en javascript). Désormais, la variable **$_POST** contient ce que nous voulons, un peu comme ce fut le cas pour la variable **$_GET**

- Désormais, en vous inspirant du TD précédent 🙂, vérifiez si la clé “action” est bien présente dans la variable **$_POST**. Si tel est le cas, et que sa valeur est égale à “delete”, alors appelez la fonction **delete()** qui accepte deux arguments : la connexion et l’identifiant de l’article issu de la variable **$_POST**.

Il ne nous reste plus qu’à créer cette fonction **delete()** !

## Serveur – Suppression effective

La fonction **delete()** sert à effectuer la suppression réelle de l’article à partir de sa clé primaire, en l’occurrence ici, son identifiant. Nous avons simplement besoin de supprimer l’élément puis de renvoyer la liste des éléments présents dans la base de données. Ainsi notre code javascript se chargera d’afficher de nouveau son contenu, à l’aide des différentes fonctions déjà créées précédemment mais également lors des TDs précédents. Pratique n’est-ce pas ? 😀

Pour cela :

- Créez la fonction **delete()** en PHP… évidemment 🙂
- Cette fonction doit accepter deux arguments : l’objet de connexion déjà instancié ainsi que l’identifiant de l’article à supprimer
- À l’aide de vos connaissances en préparation et exécution de requêtes SQL en PHP (vu précédemment et en CM 😉), adaptez et exécutez la requête SQL suivante : `"DELETE FROM vetements WHERE id = 42 ;"`. Bien évidemment, 42 est ici un identifiant donné en exemple.
- Appelez la fonction **findAll()** pour finaliser le renvoi du contenu de la BDD (Base De Données).

Ô mirrrraaaacle ! 🎉 Vous pouvez ENFIN visualiser et tester le résultat ! Allez dans votre navigateur et supprimez l’élément qui vous paraît le plus moche ! 🤭

Ne trouvez-vous pas cela magnifique ?
Bon, si on en supprime trop il ne reste plus rien. Il serait bon de pouvoir ajouter des éléments pour contrebalancer.

# Ajout d’un article

Avec la suppression vous avez pu voir votre première requête POST et faire le lien entre le client (JS) et le serveur (PHP) tout en gardant une réécriture de l’affichage. Désormais, avec l’ajout d’un article, le concept est le même à l’exception de l’ajout d’un formulaire et de son usage en asynchrone.
La nouveauté de cette fonctionnalité est donc plus à chercher du côté de Javascript que de PHP (ou la logique reste la même).

Commençons par la partie client.

# Client – Création du formulaire et de son comportement

Commençons par le formulaire de notre page. Afin de vous faire gagner du temps et de ne pas entrer trop longtemps les considérations de bootstrap, voici le code HTML pur à ajouter en dessous du bandeau principal comportant le logo :

```
<div class="container">
    <form class="row align-items-center ajaxform" method="post" action="/td10/api.php">
        <div class='col-auto'>
            <input type="number" name="id" class="form-control" id="id" placeholder="Identifiant">
        </div>
        <div class='col-auto'>
            <input type="text" name="productDisplayName" class="form-control" id="productDisplayName" placeholder="Description">
        </div>
        <div class="col-auto">
            <select class="form-select" name="gender" aria-label="gender" id="gender" placeholder="Genre">
                <option selected>Genre à choisir...</option>
                <option value="Men">Masculin</option>
                <option value="Women">Feminin</option>
            </select>
        </div>
        <div class="col-auto">
            <button type="submit" class="btn btn-primary">Envoyer</button>
        </div>
        </div>
    </form>
</div>
```

Oui, vous devez le recopier ! 😂 Vous y verrez notamment le système de grille (container - row - col) ainsi que la mise en place d'un formulaire **<form>** renvoyant vers la méthode POST et vers un fichier PHP. Ce dernier renvoi est optionnel car il convient surtout de vérifier la classe .ajaxform présente sur la balise form et qui servira à attacher un écouteur d'événement sur tous les éléments ayant cette classe 🙂

Passons désormais au comportement de ce formulaire. Ici, point de fonction qui se lancera sur un onclick. Il convient d'écouter l'événement submit qui est automatiquement déclenché dans un formulaire par l'activation du bouton de type "submit". Concrètement :

- Ajoutez un écouteur sur la fin de chargement du DOM dans la page comme ceci `document.addEventListener('DOMContentLoaded', function(){`
- Dans cet écouteur, sélectionnez toutes les balises ayant la classe "ajaxform" (utilisez par exemple le **querySelector()** ). Ajoutez-y un écouteur d'événement avec addEventListener() sur l'événement "submit" et qui déclenche une fonction anonyme ayant pour argument l'événement (régulièrement nommé "e"). Si cela vous paraît

obscur, en voici un exemple connexe :

```
el.addEventListener("click", function(e){
    // youhou je fais des trucs dans
    // cette fonction anonyme qui sera
    // déclenchée sur l'événement click
    // 🥰
    // l'argument e capture les infos
    // relatives à l'événement 🤓
});
```

- Dans cette fonction anonyme, utilisez la méthode **preventDefault()** de l'événement passé en argument afin de prévenir l'exécution du submit original (car nous voulons le contrôler par de l'AJAX 😀).
- Toujours dans la fonction anonyme, créez une nouvelle instance de la classe **FormData()** à l'aide de la cible de l'événement `new FormData(e.target)` et donné la en entrée à la méthode statique **fromEntries** de la classe **Object**. Cela vous permettra de récupérer le résultat au format de tableau associatif et donc d'avoir toutes les valeurs et les clés données dans le formulaire. Mettez ce résultat dans une variable **data**. Précisons que les clés sont les attributs "name" des balises HTML et les valeurs sont l'entrée mise par l'utilisateur.
- Désormais, *yapuka* 😀 ajouter la clé "action" avec la valeur "add" dans le tableau associatif **data**. Puis, à utiliser la fonction **motif()** déjà créée.

Et voilà pour la partie client ! 👋 Passons désormais au serveur.

## Serveur – Contrôle de la requête

Alors là ce n'est pas folichon ! Il suffit d'ajouter une condition ; en plus de la vérification de la valeur "add" pour la clé "delete" que nous avions mis précédemment, il nous faut simplement ajouter également une vérification de la valeur "add" pour cette même clé.

À vous de le faire 😀! Je vous fais confiance !

## Serveur – Ajout effectif

Nous avions mis en place une fonction dans **api.php** qui permet de supprimer un article. Cette fois-ci nous voulons créer une fonction dédiée à l'ajout. Le principe est peu ou prou le même :

- Créez une fonction **add()** qui accepte deux arguments : l'objet de connexion et le tableau associatif de données

- À l'aide de vos connaissances en préparation et exécution de requêtes SQL en PHP (vu précédemment et en CM 😏), adaptez et exécutez la requête SQL suivante :

```
"INSERT INTO vetements (id, gender, productDisplayName) VALUES (42, 'Men', 'Super TShirt');"
```

  . Bien évidemment, 42, 'Men' et 'Super TShirt' sont ici donnés en exemples. Le but est de vous limiter uniquement à ces trois champs pour ce TD. L'image sera automatiquement récupérée si l'identifiant que vous mettez existe en tant que nom d'image dans votre dossier **img/vetements**
- N'oubliez pas de bien indiquer le type de donnée pour chaque argument lors de la préparation de la requête.
- Appelez la fonction **findAll()** pour finaliser le renvoi du contenu de la BDD (Base De Données).

Et voilàààà ! Essayez votre site ! Pour plus d'effet, utilisez des identifiants dont le nom existe en tant qu'image.

Pas mal n'est-ce pas ? 🥹

Certes, il nous faudrait également ajouter l'upload d'image (le "téléversement" pour faire plaisir à l'académie française 😀). Le concept y change légèrement avec notamment le changement d'entête de la requête HTTP par l'usage de "multipart/form-data" en type de contenu.

C'est bon pour ce TD ! Youhou ! Vous en êtes venu à bout !

# Liens utiles

- https://www.php.net/manual/fr/

- https://www.w3schools.com/php/default.asp
- https://emojipedia.org/