

TP3 : Développement WEB : Créer des listes liées

Les données avec PDO, Ajax et format Json

Activités du Référentiel :

A4.1.1 Proposition d'une solution applicative

A4.1.2 Conception ou adaptation de l'interface utilisateur d'une solution applicative

A4.1.7 Développement, utilisation ou adaptation de composants logiciels

A5.2.3 Repérage des compléments de formation ou d'auto-formation utiles à l'acquisition de nouvelles compétences

A5.2.4 Étude d'une technologie, d'un composant, d'un outil ou d'une méthode

Objectifs techniques:

- ✓ Connexion à une base de données en PDO
- ✓ Appréhender Ajax et la Gestion des données par JSON
- ✓ Gestion des listes déroulantes liées sans rechargement de page

Documents fournis : fichiers TP2JS.js / styles.css / page_base.class.php / index.php / recap.php (correction TP2)

Script d'insertion d'occurrences dans la Base de données : bd_slam3_tp3.sql

Suite du TP2

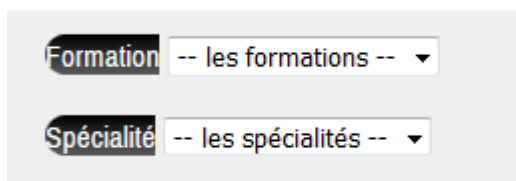
Contexte :

L'objectif ici va être d'ajouter à notre formulaire la notion de formation et de spécialités qui découlent de cette formation choisie.

Deux listes déroulantes vont donc être remplies en fonction de données issues d'une base de données.

La 1^{ère} liste contiendra toutes les formations (type BAC, BTS , DUT...).

La 2^{ème} liste sera remplie selon le choix de la 1^{ère} (Ainsi par ex, pour sélection de BAC, affichage de Bac Pro SEN, Bac STI 2D...).



L'accès aux données se fera par le biais d'une connexion PDO à une base MySQL.

Le remplissage de la 2^{ème} liste selon le choix dans la 1^{ère} liste se fera par le biais d'AjAx, sans rechargement de la page, en utilisant un objet Json.

Compréhension d'AjAx et les échanges de données

AJAX est l'acronyme d'*Asynchronous JavaScript and XML*, ce qui, transcrit en français, signifie « JavaScript et XML asynchrones ».

Derrière ce nom se cache un ensemble de technologies destinées à réaliser de rapides mises à jour du contenu d'une page Web, sans qu'elles nécessitent le moindre rechargement visible par l'utilisateur de la page Web.

Le transfert de données est géré *exclusivement* par le JavaScript, et utilise certaines technologies de formatage de données, comme le XML ou le JSON.

Tout est dit là :

<https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript/1-ajax-qu-est-ce-que-c-est>

PARTIE 1 : Mise en place de l'architecture

Dossier Class

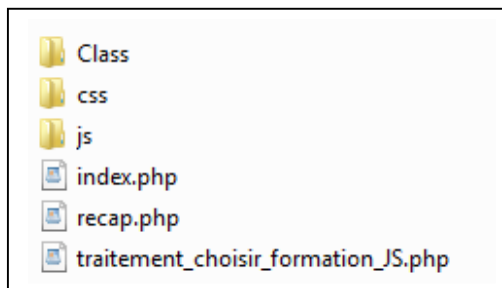
- ➔ autoload.php
- ➔ MyPDO.class.php
- ➔ Connexion.class.php
- ➔ page_base.class.php

Dossier JS (ou script)

- ➔ tous les fichiers .js

Dossier CSS (ou style)

- ➔ tous les fichiers .css



PARTIE 2 : Création de la base de données

Des exemples d'occurrences de la base de données BD_SLAM3_TP3 résumant les formations et les spécialités par formation est proposée dans le script joint : bd_slam3_tp3.sql

Travail à faire :

1. A partir de ces exemples d'occurrences, Faire le schéma conceptuel ou relationnel de cette base de données pour bien la comprendre.
2. Créer la base de données correspondante.
3. Implanter ce script d'insertion des données et compléter à votre guise les occurrences.

PARTIE 3 : Mise en place des classes

A. fichier autoload.php

4. Intérêt d'un tel fichier ?
5. Qu'impose cette fonction dans l'écriture des classes par la suite ?

```
<?php
function __autoload($nom_classe) {
    require_once $nom_classe.'.class.php';
}
?>
```

B. Mise en place de la classe MyPDO : fichier MyPDO.class.php

```
<?php
class MyPDO extends PDO
{
    public function __construct($dsn, $user=NULL, $password=NULL)
    {
        parent::__construct($dsn, $user, $password);
        //dire comment on veut traiter les erreurs ici gestion avec les exceptions try catch
        $this->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }

    //méthode exec pour les requêtes de type INSERT, UPDATE et DELETE
    public function exec($sql)
    {
        return (parent::exec($sql));
    }

    //méthode query pour les requêtes de type SELECT
    public function query($sql)
    {
        $result = parent::query($sql);
    }
}
```

```

        $result->setFetchMode(PDO::FETCH_OBJ);
//resultat de la requête retournée sous la forme d'objets
        return $result;
    }
}
?>

```

6. Quelles sont les trois fonctions indispensables ?

C. Mise en place de la classe de connexion : fichier **Connexion.class.php**

```

<?php
require_once 'MyPDO.class.php';

class Connexion{

    private $PARAM_hote='localhost';
    private $PARAM_utilisateur='root';
    private $PARAM_mot_passe='';
    private $PARAM_nom_bd='BD_SLAM3_TP3'; //nom de la base de données

    private $IDconnexion;

    public function __construct() {
        try {
            $this->IDconnexion = new MyPDO('mysql:host='.$this->PARAM_hote.';dbname='.$this->PARAM_nom_bd, $this->PARAM_utilisateur, $this->PARAM_mot_passe);
            //Il faut ajouter pour gerer les accents et caractères non utf8
            $this->IDconnexion->exec("SET NAMES 'utf8'");
        }
        catch (PDOException $e)
        {
            echo 'hote: '.$this->PARAM_hote.' '.$_SERVER['DOCUMENT_ROOT'].'<br />';
            echo 'Erreur : '.$e->getMessage().'<br />';
            echo 'N° : '.$e->getCode();
            $this->IDconnexion=false;
        }
    }

    public function __get($propriete) {
        switch ($propriete) {
            case 'IDconnexion' :
                {
                    return $this->IDconnexion;
                    break;
                }
        }
    }
}
?>

```

7. Quelles sont les actions faites dans le constructeur ici ?

D. Mise en place de la classe page base : fichier **page_base.class.php**

Reprendre votre page_base faite au TP2 ou adapter celle proposée jointe à ce TP3.

Bien mettre dans les tableaux \$style la liste de vos fichiers css et dans \$script la liste de vos fichiers js.

PARTIE 4 : Page principale contenant le formulaire et ajout des listes déroulantes

Les deux listes déroulantes vont être ajoutées à notre formulaire.
Notre formulaire est inséré dans une page principale de type « page_base ».

Reprendre ce qui a été fait en TP2 ou vous inspirez d'une correction possible donnée dans le fichier : « index.php ».

Ajouter deux listes déroulantes à votre formulaire, l'une pour les formations, l'autre pour les spécialités.
Exemple :

```
<div><label for="specialite">Spécialité</label>
  <select id="maSpe" name="maSpe" >
    <option value=" ">-- les spécialités --</option>
  </select>
</div>
```

PARTIE 5 : Connexion à la base de données pour charger les formations

➤ Connexion à la BD

On va faire appel au constructeur de la classe Connexion et lancer une requête pour lister toutes les formations.

Dans votre page index.php, créer la connexion.

Pour vous aider : Exemple d'une connexion pour récupérer l'ensemble des régions :

```
$ConnexionBaseSIO = new Connexion();
$IDconnexion = $ConnexionBaseSIO->IDconnexion;

if (!$IDconnexion){
    $pageInitiale->corps = "problème d'accès à la base de données
</article></section>";
}
else
{

    $CollRegions = $IDconnexion->query("SELECT * FROM REGION");

    // suite du code permettant de créer le formulaire dans le corps de la page
    $pageInitiale->corps .= ' bd connectée <br>
    <form method="post" ....
```

Questions :

8. Quelle est l'instruction qui permet d'exécuter la requête ?
9. Où vont être stockées toutes nos données issues de la requête ?
10. **Tester** si la connexion se passe bien.

➤ **Remplissage de la liste déroulante** : Une fois la liste déroulante des formations mise en place en HTML, nous allons la remplir en parcourant nos données issues de la requête.

Exemple de code pour remplir avec l'ensemble des régions (à adapter selon votre base de données et votre formulaire bien entendu) :

```
$pageInitiale->corps .= '
<div>
  <label for="region">Region</label>
  <select name="maRegion" id="maRegion">
    <option value=" ">-- les regions --</option>

  ' ;
```

```

foreach($CollRegions as $suneR)
{
    $pageInitiale->corps .= '
        <option value="'. $suneR->IdRegion. '>'. $suneR->Intitule. ' ' . '</option>';
    }
    $pageInitiale->corps .= '</select></div>';

    $CollRegions -> closeCursor(); // pour libérer la mémoire occupée par le résultat
de la requête
    $CollRegions = null ; // pour une autre exécution avec cette variable

```

Questions pour bien comprendre ce code:

11. Quelle est l'instruction qui parcourt toutes les données liées à la table REGION ?
12. Comment je fais pour atteindre un intitulé précis d'une region (quelle variable est utilisée) ?
13. Où est la ligne qui remplit ma liste déroulante à partir de ces données ?
14. Ma 2^{ème} liste déroulante des spécialités est-elle remplie ici ? pourquoi ?

➤ **TESTER** si la connexion se passe bien et la liste se remplit.

➤ **PARTIE 6 : Mise en place de la fonction ajax qui déclenchera la sélection des spécialités au choix de la formation**

Fichier jsFormation.js

A bien inclure dans la liste des fichiers script dans la classe page_base.

15. A partir de l'exemple suivant qui permet de choisir un département à partir d'une région, créer votre fichier « jsFormation.js » :

```

$(document).ready(function() {
    console.log( "ready!" );

    // à la sélection d une formation dans la liste
    $('#maRegion').change(function(){
        var val = $(this).val(); // on récupère la valeur de la région
        console.log("numéro de la région : "+ val);

        if(val != " ") {
            $('#monDept').empty(); // on vide la liste des départements

            var filterDataRequest = $.ajax({
                url: 'traitement_choisir_Region_JS.php',
                type: 'GET',
                data: 'IdRegion='+ val, // on envoie $_GET['IdRegion']
                dataType: 'json'
            });
            filterDataRequest.done(function(data) {
                console.log("success");
                console.log(data);
                $.each(data, function(index, value) {
                    $('#monDept').append('<option value="'+ index +' ">'+ value
+ '</option>');
                });
            });
            filterDataRequest.fail(function(jqXHR, textStatus) {
                console.log( "error" );
                if (jqXHR.status === 0){alert("Not connect.n Verify Network.");}
                else if (jqXHR.status == 404){alert("Requested page not found.
[404]");}
                else if (jqXHR.status == 500){alert("Internal Server Error
[500].");}
                else if (textStatus === "parsererror"){alert("Requested JSON
parse failed.");}
                else if (textStatus === "timeout"){alert("Time out error.");}
                else if (textStatus === "abort"){alert("Ajax request
aborted.");}
            });
        }
    });

```

```

        else{alert("Uncaught Error.n" + jqXHR.responseText);}
    });
    filterDataRequest.always(function() {
        console.log( "complete" );
    });
} // fin du if val est vide
});
});

```

16. Quand se déclenche ce script JavaScript ?

17. A quoi sert la fonction `var filterDataRequest = $.ajax(...)` ?

18. A quoi sert la fonction `filterDataRequest.fail(function(jqXHR, textStatus) ?`

19. Que doit-il se passer ensuite (suite au déclenchement de ce script JavaScript) ?

➤ PARTIE 7 : Mise en place du traitement pour récupérer les spécialités selon la formation

Fichier `traitement_choisir_formation_js.php`

20. A partir de l'exemple suivant qui permet de sélectionner les départements à partir de la région sélectionnée, créer votre fichier « `traitement_choisir_formation_js.php` » :

```

<?php
require_once(' ./Class/Connexion.class.php');

if(isset($_GET['IdRegion'])) {

    $json = array();

    // on récupère l'Id de la région sélectionnée
    $idR = htmlentities(intval($_GET['IdRegion']));

    // requête qui récupère les départements selon la région
    $requete = "SELECT IdDpt, Libelle FROM Departement WHERE idRegion= ".
    $idR."; " ;

    // exécution de la requête
    $newConnexion = new Connexion();
    $idD = $newConnexion->IDconnexion;
    $resultat = $idD->query($requete);

    foreach($resultat as $donnees) {
        $valeur = $donnees->Libelle;

        $json[$donnees->IdDpt] = utf8_encode($valeur);
    }
}

// envoi du résultat au success
echo json_encode($json);

?>

```

21. A quoi sert le 1^{er} if ?

22. Que permet de récupérer la requête ?

23. Que met-on dans `$valeur` ?

24. Que met-on dans `$json` ?

➤ TESTER.

25. Qu'est-ce qui change sur la page web du formulaire à chaque changement de choix de formation ?