

TP6 – Création d'un objet

Pour le moment, nous pouvons consulter les données de la base et les afficher, mais nous ne pouvons les modifier ou en créer de nouvelles que par l'interface phpMyadmin.

Dans ce TP nous allons mettre en place des formulaires de création et de modification pour avoir une interface dédiée à la création et à la modification directement sur l'application web.

Exercice 1 – Le formulaire de création d'un Auteur

Dupliquez votre TP5/ex5 en TP6/ex1.

1. Créez dans TP5/ex1/vue un fichier `formulaireCreationAuteur.html` dont le code sera le suivant :

```
formulaireCreationAuteur.html X
1  <form action="routeur.php" method="get">
2    <input type="hidden" name="controleur" value="ControleurAuteur">
3    <input type="hidden" name="action" value="creerAuteur">
4    <div>
5      <label for="nom">nom</label>
6      <input type="text" name="nom" placeholder="nom" required>
7    </div>
8    <div>
9      <label for="prenom">prénom</label>
10     <input type="text" name="prenom" placeholder="prénom" required>
11   </div>
12   <div>
13     <label for="anneeNaissance">année denaissance</label>
14     <input type="number" name="anneeNaissance" placeholder="année denaissance" required>
15   </div>
16   <button type="submit">créer</button>
17 </form>
```

Remarques :

- La méthode du formulaire est `get` et le restera tout le temps de production du site. En effet les informations recueillies dans le formulaire transitent alors par l'url et sont donc visibles ce qui nous aidera à bien comprendre nos éventuelles erreurs ;
- Chaque champ à compléter est dans un `<div>`, se compose d'un `<label>` qui lui est attaché. Chaque champ est requis. Le `placeholder` permet de suggérer la réponse ;
- Il y a deux `<input>` particuliers. Ils sont de type « `hidden` ». C'est nécessaire car si le formulaire est en « `get` », toutes les paires clé/valeur à transmettre sont issues des `input` du formulaire seulement.

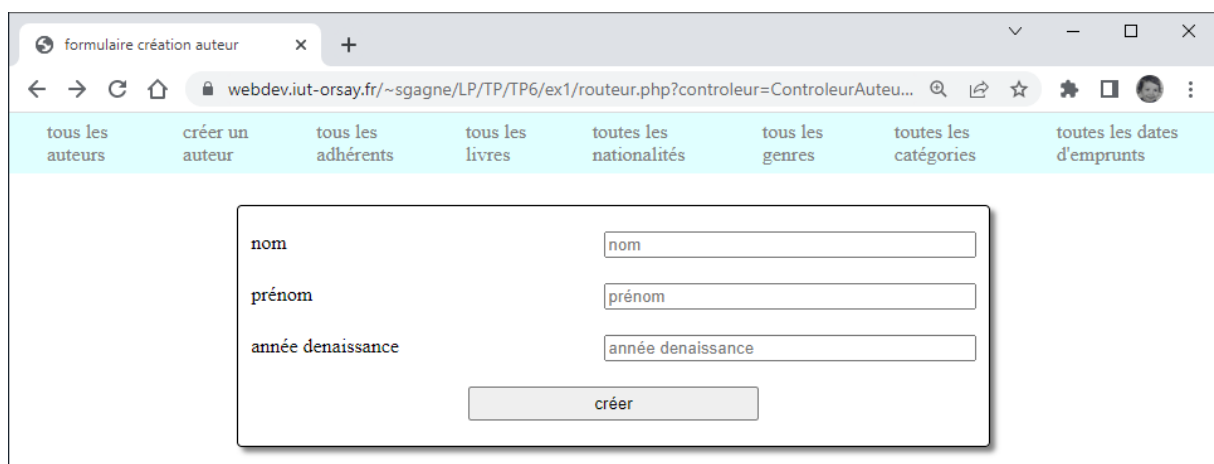
En particulier, une `action="routeur.php?controleur=..."` ne transmettrait pas les informations `controleur=...` D'où ces deux champs de type `hidden`.

2. Créez, dans le `controleurAuteur`, une méthode

```
public static function afficherFormulaireCreationAuteur() {  
  
}
```

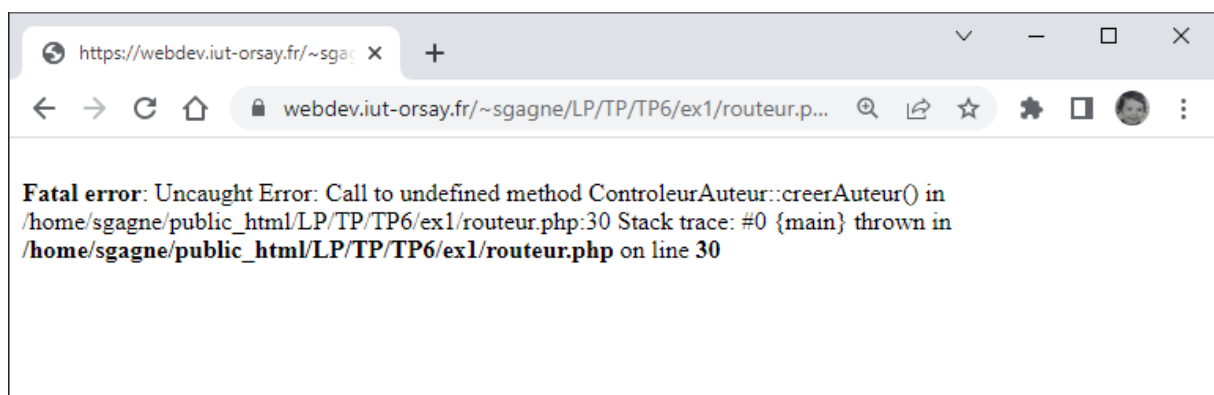
qui donnera une valeur à `$titre`, puis affichera les vues classiques, ainsi que la vue du formulaire.

3. Incorporez dans le menu un lien qui permet d'afficher le formulaire.
4. Transférez, vérifiez que tout fonctionne.



En particulier, remplissez le formulaire et vérifiez que l'url de passage vers le traitement du formulaire contient bien toutes les informations.

5. Pourquoi une erreur se produit-elle ?



Exercice 2 – Les méthodes `addAuteur` de `Auteur` et `creerAuteur` de `ControleurAuteur`

Dupliquez TP6/ex1 en TP6/ex2.

1. En utilisant une requête préparée, codez dans la classe `Auteur` la méthode

```
public static function addAuteur($n,$p,$a) {  
  
}
```

Cette méthode retournera `true` si l'insertion s'est bien passée, et `false` sinon. Pour les auteurs, voici le code à produire. Pour les autres classes, vous adapterez :

```
// méthode d'insertion  
public static function addAuteur($n,$p,$a) {  
    $requetePrepree = "INSERT INTO Auteur(`nom`,`prenom`,`anneeNaissance`) VALUES(:n_tag,:p_tag,:a_tag);";  
    $req_prep = Connexion::pdo()->prepare($requetePrepree);  
    $valeurs = array("n_tag" => $n,"p_tag" => $p,"a_tag" => $a);  
    try {  
        $req_prep->execute($valeurs);  
        return true;  
    } catch(PDOException $e) {  
        return false;  
    }  
}
```

2. Créez, dans la classe `ControleurAuteur`, une méthode

```
public static function creerAuteur() {  
  
}
```

Cette méthode :

- donne sa valeur à `$titre`,
- récupère du tableau `$_GET` les nom, prénom et année de naissance de l'auteur à créer,
- appelle la méthode `addAuteur` du modèle en récupérant le résultat booléen,
- si le booléen est `true`, affiche la liste des auteurs, et sinon, réaffiche le formulaire de création.

3. Testez l'insertion, vérifiez que tout fonctionne.

Exercice 3 – La même démarche pour les adhérents

Dupliquez TP6/ex2 en TP6/ex3.

Reproduisez exactement les étapes des exercices 1 et 2 pour créer une fonctionnalité de création d'un adhérent.

Remarques importantes

Le travail sur la création d'un auteur puis d'un adhérent a été proposé pour se rendre compte de plusieurs choses :

- Les deux formulaires ont la même structure, juste un nombre différents de lignes (chaque ligne étant représentée par une `<div>` ;
- Même remarque pour les deux méthodes `addAuteur` et `addAdherent` ;
- Même remarque pour les méthodes `creerAuteur` et `creerAdherent`.

Dans la logique de généricité, nous allons reprendre le travail précédent pour faire apparaître :

- Un formulaire commun `formulaireCreationObjet.php`, qui utilisera pour afficher ses lignes un attribut `static` de la classe contrôleur concernée ;
- Une méthode générique `addObjet` de la classe `Objet`;
- Une méthode générique `creerObjet` de `ControleurObjet`.

Exercice 4 – Générisation du formulaire de création

Dupliquez TP6/ex3 en TP6/ex4.

1. Si vous observez les deux formulaires de création actuels, ils ne diffèrent que par le nombre de champs à compléter, le type de ces champs, leur nom et leur nom apparent. Par exemple :
 - `anneeNaissance` est de type `number` et ce champ a pour nom apparent « année de naissance »,
 - `mdp` est de type `password` et a pour nom apparent « mot de passe »,
 - `email` est de type `email` et a pour nom apparent « email ».

Nous allons donc structurer, pour les deux classes `ContrôleurAdherent` et `ContrôleurAuteur`, un attribut static `$tableauChamps` qui renseigne sur ces noms, types et noms apparents.

Par exemple, pour les auteurs, le début de code du `ContrôleurAuteur` sera :

```
require_once("modele/auteur.php");

class ContrôleurAuteur extends ContrôleurObjet {

    // attributs de classe
    protected static $objet = "Auteur";
    protected static $cle = "numAuteur";
    protected static $tableauChamps = array(
        "nom" => ["text", "nom"],
        "prenom" => ["text", "prénom"],
        "anneeNaissance" => ["number", "année de naissance"]
    );
}
```

Mettez en place cet attribut pour la classe `ContrôleurAuteur`.

2. Codons maintenant le formulaire de création générique :

a. Si on compare le début des deux formulaires :

```
<form action="routeur.php" method="get">
    <input type="hidden" name="contrôleur" value="ContrôleurAuteur">
    <input type="hidden" name="action" value="creerAuteur">

<form action="routeur.php" method="get">
    <input type="hidden" name="contrôleur" value="ContrôleurAdherent">
    <input type="hidden" name="action" value="creerAdherent">
```

Les seules différences sont au niveau des mots `Auteur` et `Adherent`, qui est récupérable par `static::$objet`.

Créez le fichier `formulaireCreationObjet.php` ayant pour début ces trois lignes, en remplaçant les mots `Auteur` et `Adherent` par une instruction

```
<?php echo $table; ?>
```

Ce sera bien sûr au `ContrôleurObjet` de définir la valeur de `$objet`. Si vous avez compris, cette valeur sera `$table = static::$objet;`

- b. Si on observe les différentes balises `<div>` composant le corps du formulaire, elles sont toutes structurées de la même façon :

```
<div>
  <label for="prenom">prénom</label>
  <input type="text" name="prenom" placeholder="prénom" required>
</div>
```

Le `for` et le `name` correspondent au nom du champ dans le tableau `static` des champs, le contenu visible du `label` et le `placeholder` sont le nom apparent. Le `type` est l'attribut `type`.

Au moyen d'une boucle `foreach` agissant sur le tableau des champs, créez l'ensemble des balises `<div>` du formulaire.

- c. Terminez en ajoutant le bouton « créer ».
- d. Modifiez le menu pour que le lien « créer un auteur » donne maintenant comme action `afficherFormulaireCreationObjet`.
- e. Codez la méthode `afficherFormulaireCreationObjet` de `ControleurObjet` qui récupère tous les attributs `static` nécessaires et qui demande l'affichage du formulaire.
- f. Supprimez le fichier `formulaireCreationAuteur.html`, transférez et vérifiez que l'appel au formulaire de création d'un auteur est fonctionnel. Testez la création d'un auteur. Vous ferez attention que dans le cas d'une erreur de création, on retourne sur le formulaire de création générique maintenant !
3. Reprenez exactement les mêmes démarches pour la création d'un adhérent, en particulier le tableau `static` des champs dans `ControleurAdherent`. N'oubliez pas de changer le menu, et de supprimer les fichiers de création devenus inutiles.

Exercice 5 – Adaptation des classes `Controleur` au formulaire générique

Dupliquez TP5/ex4 en TP5/ex5.

Dans cet exercice, on fait en sorte que chaque classe puisse avoir accès au formulaire de création générique.

1. Dans le menu, ajoutez des liens vers le formulaire de création générique pour les livres, les nationalités et les catégories. Pour le moment on laisse les dates d'emprunt de côté. Il pourrait d'ailleurs être opportun de structurer un peu mieux le menu, mais c'est laissé à votre initiative...
2. Dans chaque classe Modèle, codez sur le modèle de `addAuteur` les différentes méthodes `addLivre`, `addNationalite`, `addGenre` et `addCategorie`.
3. Dans les divers contrôleurs, ajoutez les attributs `static $tableauChamps` qui permettront de fabriquer le formulaire correspondant à la classe à partir du formulaire générique.
4. Créez de même, dans chaque contrôleur, les méthodes `creerLivre`, `creerNationalite`, ...
5. Transférez, et vérifiez que tout fonctionne en créant une catégorie, un genre, un livre, une nationalité.

Dans le TP suivant nous allons mettre en place, avec le plus de généricité possible, les fonctionnalités de modification et de suppression d'un objet.