TP9 – Création d'objets (difficile)

Contexte et objectifs :

Nous allons, dans ce TP9, coder les fonctionnalités de création d'objets. Ces fonctionnalités seront appelées par des formulaires vides. Dans un souci de généricité, nous ferons en sorte de développer ces fonctionnalités directement dans les classes mères.

Il y aura donc un aspect abstrait, et ceci nécessitera toute votre attention.

Les formulaires sont construits par les contrôleurs, car ce sont des formulaires qui nécessitent de la logique : ils ne présentent pas a priori le même nombre de champs en fonction des objets. Ils feront donc l'objet d'une constructions précise et bien pensée.

Ces champs (ou ces attributs, cela revient au même) devront être rapidement identifiables par la classe mère. C'est pourquoi nous allons construire, pour chaque classe fille, un tableau static des champs concernés.

Nous allons développer tout cela pour les adhérents en premier lieu, puis nous généraliserons autant que possible. La généricité est simple pour toutes les classes qui, comme adherent, n'ont pas de dépendance : adherent, nationalite, etat et categorie.

Dupliquez votre code précédent dans TP9/ex1.

Exercice 1 – Le formulaire de création d'un adhérent

Quand nous allons créer le formulaire de création d'un adhérent, certains champs sont de type « text », un autre de type « password », un autre de type « email ». Voici les correspondances :

Nom du champ	Type de l'input	Contenu du label correspondant
login	text	identifiant
mdp	password	mot de passe
nomAdherent	text	nom
prenomAdherent	text	prénom
email	email	email
telephone	text	téléphone

Le nom du champ est rarement égal au contenu du label dans le formulaire (espaces, accents, etc). Ces informations doivent donc être données par le contrôleur concerné.

Par exemple, pour le controllerAdherent, nous allons construire un attribut static :

```php
protected static $champs = array(
  "login" => ["text", "identifiant"],
  "mdp" => ["password", "mot de passe"],
  "nomAdherent" => ["text", "nom"],
  "prenomAdherent" => ["text", "prénom"],
  "email" => ["email", "email"],
  "telephone" => ["text", "téléphone"]
);
```

Chaque clé de ce tableau désigne l'un des champs. La valeur correspondante est un tableau donnant le type du champ en entrée 0, et le contenu du label correspondant en entrée 1. Ainsi :

- `$champs["login"][0]` donne le type (ici « `text` »),
- `$champs["login"][1]` donne le contenu du label (ici « `identifiant` »).

Par une lecture en boucle de ce tableau `$champs`, nous pourrons donc construire le formulaire.

1. Dans le fichier `controllerAdherent.php`, créez l'attribut `static $champs` comme indiqué ci-dessus.

2. Dans le fichier `controllerObjet.php`, créez la méthode suivante :

```php
public static function displayCreationForm() {
  $champs = static::$champs;
  $classe = static::$classe;
  $identifiant = static::$identifiant;
  $title = "création ".$classe;
  include("view/debut.php");
  include("view/menu.html");
  include("view/formulaireCreation.php");
  include("view/fin.php");
}
```

Cette méthode récupère tout ce qu'il y a à récupérer de la classe contrôleur fille, construit le titre, puis convoque les vues classiques, plus une vue pas encore créée qui contiendra le formulaire construit à partir de l'attribut `static $champs` récupéré auprès de la classe contrôleur fille.

3. Nous allons maintenant créer le formulaire dans un fichier `formulaireCreation.php`. Ce formulaire a quelques particularités :

   a. Il fonctionne en méthode `get`, et l'action sera `routeur.php`. Rien d'autre. En effet, nous allons passer dans l'url (donc avec la méthode `get`) les valeurs du formulaire, et dans ce cas nous ne pouvons pas, dans l'action de la balise `form`, indiquer quoi que ce soit. Une balise de cette forme serait une erreur :

```html
<form action="routeur.php&objet=adherent&action=create" method="get">
```

Vous utiliserez donc une balise `form` de ce type :

```html
<form action="routeur.php" method="get">
```

   b. Par contre, il faut trouver le moyen de communiquer l'objet et l'action. Cela se fera au moyen de deux balises input de type

« hidden » qui n'apparaissent pas dans le rendu de la page web :

```
<form action="routeur.php" method="get">
  <input type="hidden" name="objet" value="<?php echo $classe; ?>">
  <input type="hidden" name="action" value="create">
```

Vous pouvez observer que la valeur de la balise « objet » reprend la valeur de l'attribut `static $classe` fournie par la classe contrôleur fille. Elle s'adaptera donc à adherent, bd, etat, ...

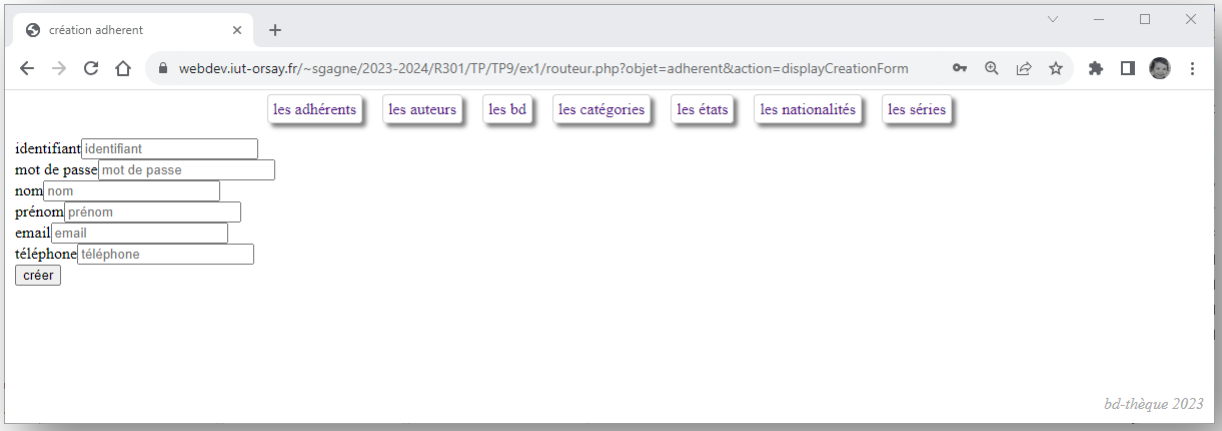
c. Il faut maintenant créer une ligne de formulaire pour chaque champ du tableau `$champs`. Pour cela, il faut parcourir ce tableau dans une boucle `foreach`. Pour chaque champ, il y a un `label` avec un contenu déjà évoqué, et un `input` avec un `type`, un `name` et un `placeholder` à préciser. Cela peut finalement donner un code comme le code suivant :

```
<main>
  <form action="routeur.php" method="get">
    <input type="hidden" name="objet" value="<?php echo $classe; ?>">
    <input type="hidden" name="action" value="create">
    <?php
      foreach ($champs as $champ => $details) {
        echo "<div>";
        echo "<label for=\"$champ\">$details[1]</label>";
        echo "<input type=\"$details[0]\" name=\"$champ\" placeholder=\"$details[1]\" required>";
        echo "</div>";
      }
    ?>
    <button type="submit">créer</button>
  </form>
</main>
```

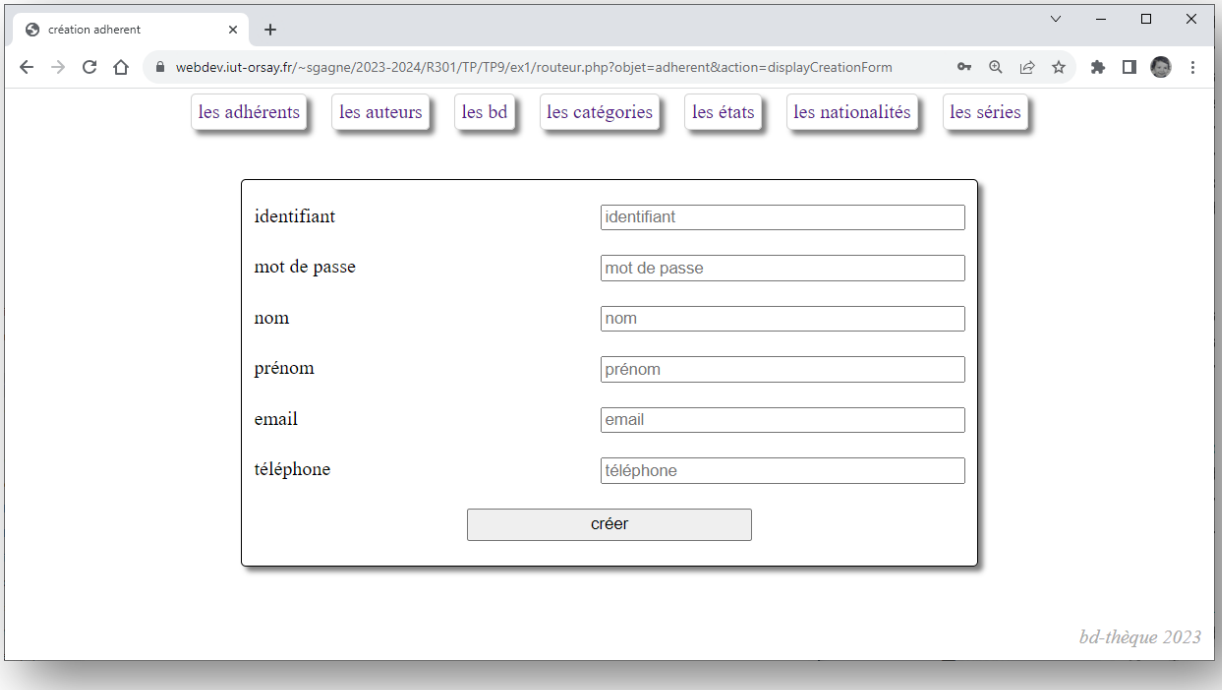
Il est important de bien comprendre l'intégralité de ce code, sa généricité et la manière de traiter les informations de `$champs`. Créez ce formulaire sans oublier le `button submit`.

4. Dans le routeur, ajoutez l'action displayCreationForm. Testez le bon fonctionnement en appelant l'url

`TP9/ex1/routeur.php?action=displayCreationForm&objet=adherent`



5. C'est très laid n'est-ce pas ? A vous d'améliorer tout ça en css. Par exemple :



6. Si vous remplissez le formulaire, et si vous envoyez les données, pourquoi voyez-vous s'afficher la liste des adhérents ?

Exercice 2 – La création d'un adhérent

Dupliquez TP9/ex1 en TP9/ex2.

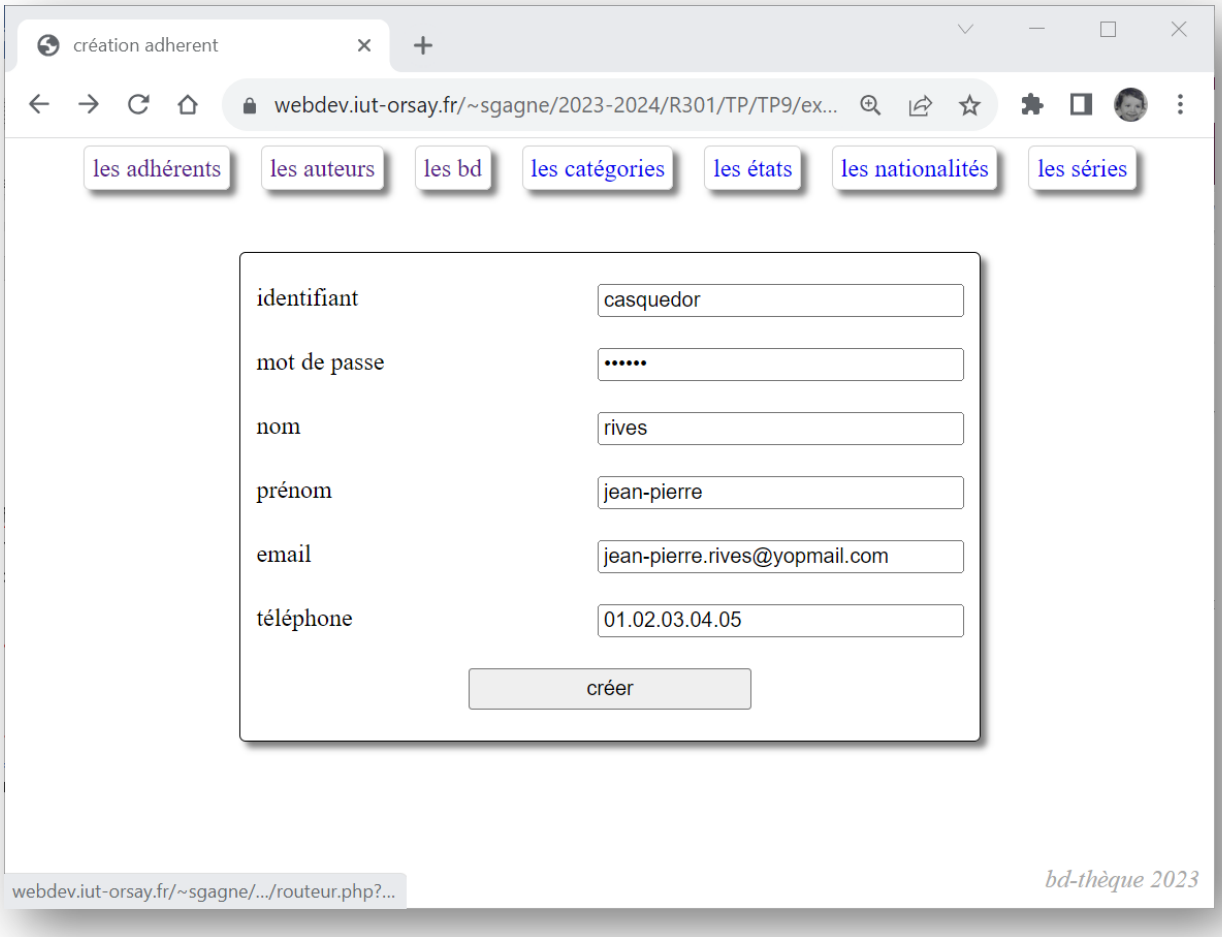
La création d'un objet (par exemple un adhérent) va se faire selon les étapes suivantes :

- Le formulaire de création de l'exercice précédent envoie les données en lançant la méthode create du contrôleur dédié,
- Ce contrôleur récupère les données envoyées, et appelle la méthode create du modèle cette fois, avec les données en question,
- Le modèle, à partir des données reçues, construit la requête préparée de création,
- Et enfin le contrôleur demande l'affichage de la liste.

1. On commence par la méthode create du contrôleur controllerObjet : créez la méthode public static function create() qui :

   - Récupère l'attribut static $champs de la classe contrôleur fille,
   - Crée un tableau vide $donnees (qui sera utilisé par la classe modèle),
   - Parcourt, dans une boucle foreach, le tableau $_GET. Pour chaque clé de $_GET qui n'est égale ni à "objet" ni à "action", on insère le couple clé / valeur dans $donnees. Ceci a pour effet de ne pas prendre de $_GET les entrées "objet" et "action" des input de type hidden.
   - Appelle la méthode create du modèle (pas encore écrite) en lui passant en paramètre le tableau $donnees,
   - Enfin, appelle la méthode displayAll.

2. Maintenant la méthode create du fichier objet.php. On va créer la méthode public static function create($donnees) dont l'objectif essentiel est de structurer correctement la requête d'insertion avant de l'exécuter.
   Prenons un exemple. On remplit le formulaire de création d'un adhérent :



Alors, quand on soumettra ce formulaire, voilà ce qui sera transmis par l'url (tableau $_GET) et ce qui est fabriqué par le contrôleur (tableau $donnees) pour être utilisé par la requête :

```
$_GET :Array
(
    [objet] => adherent
    [action] => create
    [login] => casquedor
    [mdp] => 123456
    [nomAdherent] => rives
    [prenomAdherent] => jean-pierre
    [email] => jean-pierre.rives@yopmail.com
    [telephone] => 01.02.03.04.05
)

$donnees :Array
(
    [login] => casquedor
    [mdp] => 123456
    [nomAdherent] => rives
    [prenomAdherent] => jean-pierre
    [email] => jean-pierre.rives@yopmail.com
    [telephone] => 01.02.03.04.05
)
```

La différence entre les deux vient du fait que pour construire $donnees, on a écarté les entrées objet et action.

Grâce à $donnees, la méthode create de la classe mère objet va structurer ce type de chaîne de caractères :

```
$requetePreparee = "INSERT INTO adherent(`login`,`mdp`,`nomAdherent`,`prenomAdherent`,`email`,`telephone`)
VALUES(:login,:mdp,:nomAdherent,:prenomAdherent,:email,:telephone);"
```

Le nom de la table (adherent) est facilement récupérable dans la classe fille adherent, le nom des champs est récupérable dans les clés de $donnees, et le nom des différents tags de la requête préparée a aussi été fabriqué à partir des clés de $donnees.

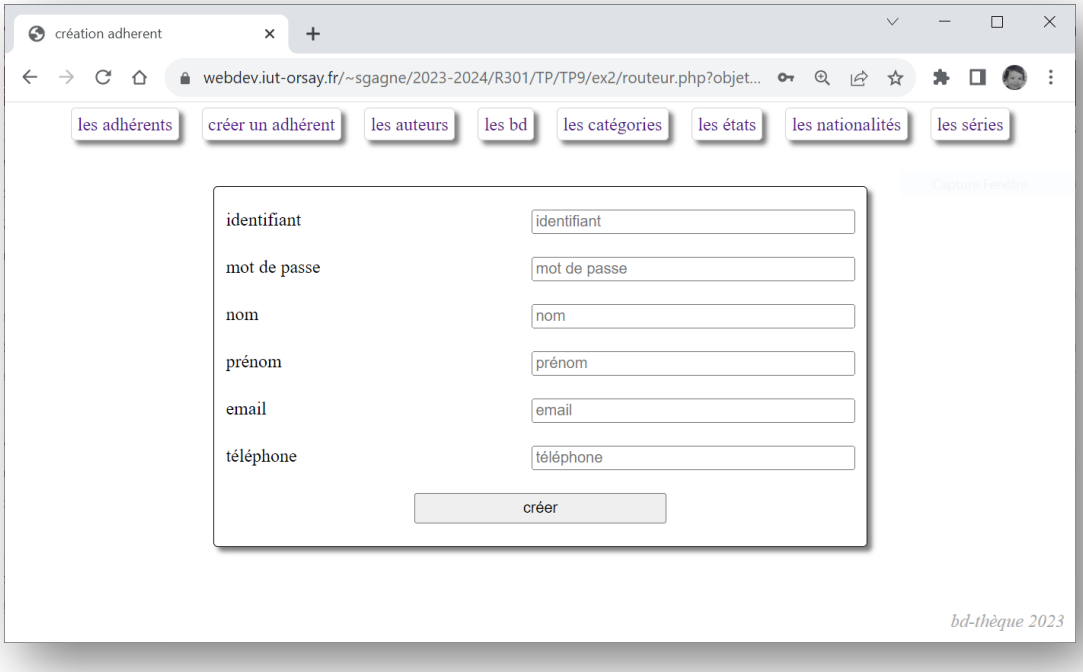
Il y a donc moyen de fabriquer facilement cette chaîne de caractères, grâce à des boucles.

Ensuite, au moment de lancer la méthode execute de la classe PDO, il faudra penser au tableau des valeurs correspondant à chaque tag. Si

votre construction est cohérente, ce tableau est tout simplement $donnees...

Voilà, vous avez le plan. A vous de créer la méthode public static function create($donnees) de la classe mère objet. COURAGE !

3. Pour faire facilement apparaître le formulaire de création d'un adhérent, créez un lien vers le formulaire dans le menu. Vous pouvez même compacter le menu avec des sous-menus si vous voulez.



4. Remplissez le formulaire et envoyez-le. Si tout se passe bien, vous n'avez pas d'erreur, mais pas non plus d'adhérent créé. Trouvez ce qui n'a pas été ajouté au routeur, et vérifiez que maintenant vous pouvez ajouter un nouvel adhérent !

Exercice 3 – Généralisation du protocole de création

Dupliquez TP9/ex2 en TP9/ex3.

Dans cet exercice, vous devez, en autonomie, dupliquer le processus de création aux objets suivants : nationalite, etat et categorie. Ce sont les seuls objets, avec adherent, à posséder une structure « indépendante », à la différence de serie par exemple, qui dépend de categorie.

Ces objets ne nécessitent pas dans le formulaire un champ relatif à leur identifiant (car ils sont en AUTO_INCREMENT dans la base). Donc l’attribut static qui sert à la construction de ce formulaire doit être légèrement adapté…

Mettez tout en place pour créer des nationalités, des états et des catégories. Testez.

Les objets complexes que sont serie, bd et auteur feront l’objet d’un traitement à part pour la construction du formulaire, dans le TP suivant. En effet, certains champs sont des clés étrangères, et dans un formulaire, un identifiant pur et dur n’est pas simple à interpréter.