TP13 – Les emprunts

On va maintenant aborder le dernier aspect métier : les emprunts de livres par des adhérents.

Dupliquez votre code `TP12/ex7` en `TP13/ex1`.

Exercice 1 – Disponibilité d'un livre

Un livre est indisponible (donc actuellement emprunté) s'il figure dans une ligne de la table emprunte.

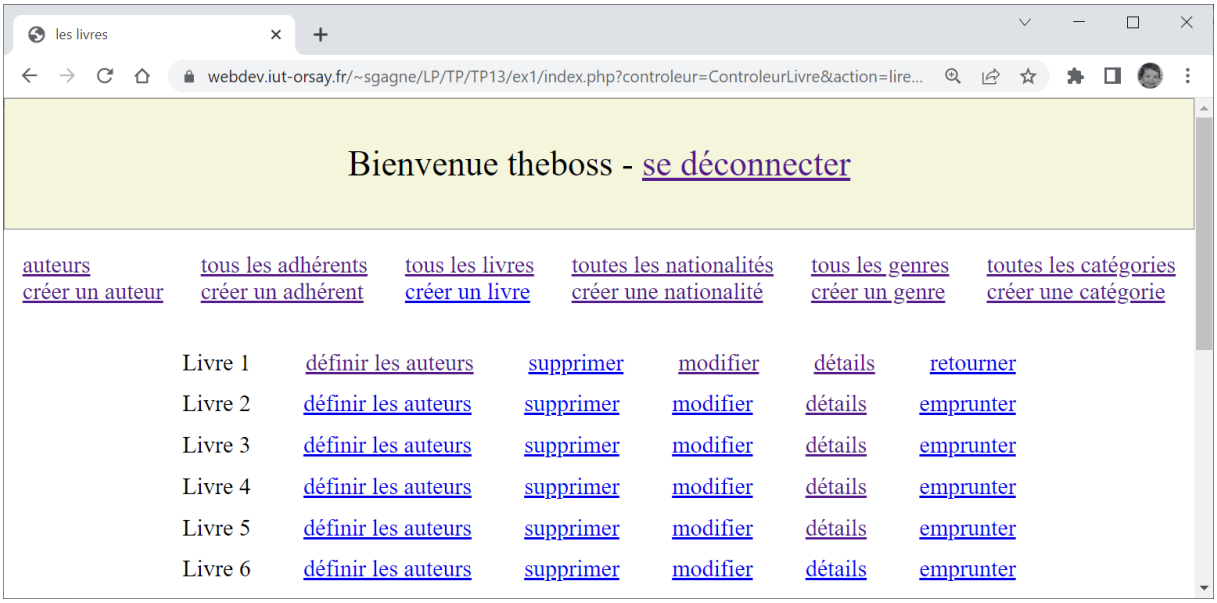
1. Créez, dans la classe Livre, une méthode d'instance

   ```
   public function estDisponible()
   ```

   qui retourne vrai si le livre courant n'est pas dans la table `emprunte`.
   Vous pourrez, par exemple, récupérer le tableau des `numLivre` de la table `emprunte`, et vérifier si le `numLivre` du livre courant est dans ce tableau.

2. Modifiez la méthode `afficher` d'un `Livre` en précisant dans le descriptif si le livre est disponible.

3. On convient que chaque adhérent peut savoir si un livre est ou non disponible, par contre seul l'administrateur peut lancer les actions « emprunter » ou « retourner » un livre. On va donc modifier, dans le `ControleurObjet`, la méthode `lireObjets`, pour afficher soit un lien « emprunter », soit un lien « retourner » selon le cas :



- Le lien « emprunter » apellera le `ControleurLivre` avec comme action `afficherFormulaireEmpruntLivremprunterLivre`, en fournissant le `numLivre` comme identifiant ;

- Le lien « retourner » appellera l'action `retournerLivre` du `ControleurLivre`, en fournissant le `numLivre` comme identifiant ;

  Ces actions ne sont pas encore codées, et feront l'objet d'un exercice ultérieur.

Exercice 2 – Capacité d'emprunt d'un adhérent

On sait maintenant si un livre est disponible ou non. Il va falloir parallèlement à ça coder des méthodes qui disent si oui ou non un adhérent a une capacité d'emprunt (s'il n'a pas déjà atteint son nombre d'emprunts simultanés autorisé).

Dupliquez votre code `TP13/ex1` en `TP13/ex2`.

1. Dans la classe `Adherent`, créez une méthode d'instance

```
public function nbEmprunts()
```

qui retourne le nombre d'emprunts en cours pour l'adhérent courant.

2. Dans la classe `Adherent`, créez une méthode d'instance

```
public function getCapaciteEmprunt()
```

qui retourne le nombre total d'emprunts simultanés autorisé pour l'adhérent courant.

3. Dans la classe `Adherent`, créez une méthode d'instance

```
public function peutEmprunter()
```

qui retourne vrai si le nombre total d'emprunts de l'adhérent est inférieur à sa capacité totale d'emprunt, et faux sinon.

4. Toujours dans la classe `Adherent`, Modifiez aussi la méthode afficher d'un adhérent pour qu'on puisse voir combien il a d'emprunts en cours, quelle est sa capacité totale d'emprunts et combien il peut encore emprunter de livres, avec un bilan sur sa capacité à emprunter.

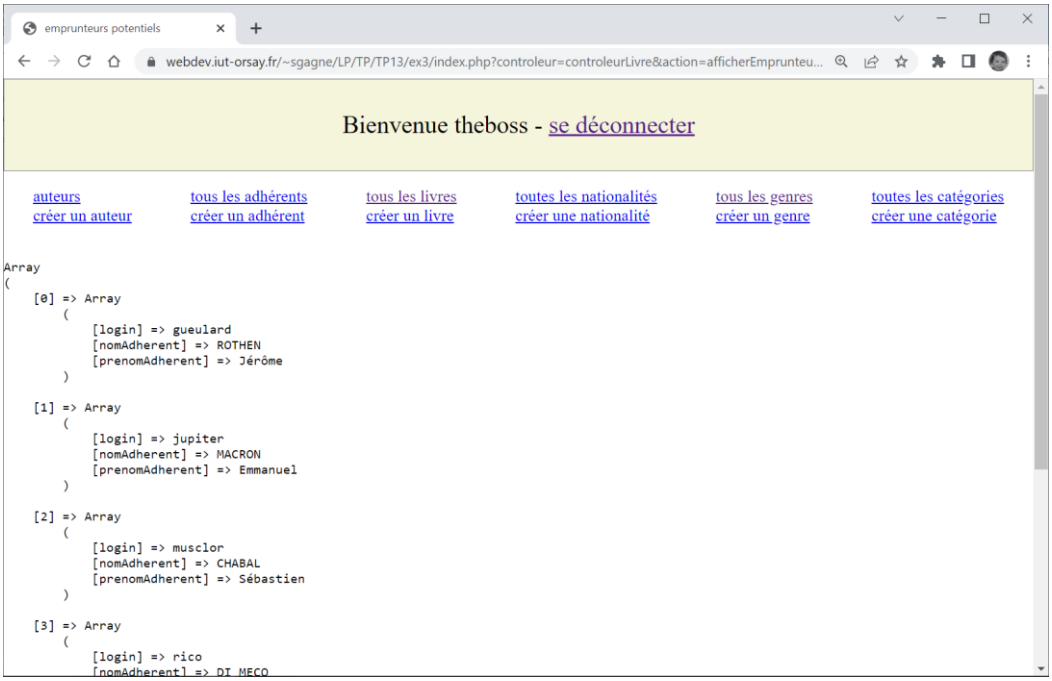
Exercice 3 – Les emprunteurs potentiels

Dupliquez votre code `TP13/ex2` en `TP13/ex3`.

1. Créez dans la classe `Adherent` une méthode de classe

```
public static function getEmprunteursPotentiels()
```

qui retourne un tableau des adhérents qui peuvent encore emprunter :



2. Créez l'action `afficherFormulaireEmpruntLivre` :

Si l'utilisateur connecté est admin :

- On récupère le `numLivre` du `$_GET` ;
- On récupère le titre du livre ;
- On calcule le tableau des emprunteurs potentiels ;
- On fabrique, à partir de ce tableau, une balise `select` où chaque `option` affiche le nom et le prénom du potentiel emprunteur, mais garde comme `value` le login de cet adhérent ;
- On insère les vues `debut.php`, le menu, la vue pas encore codée et qui s'appellera `vue/formulaireEmpruntLivre.php`, et enfin la vue `fin.html` ;

Sinon :

- On affiche la liste des objets.

3. Créez la vue `formulaireEmpruntLivre.php` qui :

   - Présente 3 balises input de type « `hidden` » : une pour le contrôleur (valeur `ControleurLivre`), une pour l'action (valeur `emprunterLivre`) et une pour le `numLivre` (valeur le `numLivre` récupéré du `$_GET` par le `controleurLivre`),

   - Affiche le titre du livre et la balise `select` des emprunteurs potentiels.

   Voici une capture possible :

   Le clic sur « valider l'emprunt » appelle donc l'action `emprunterLivre`. Nous allons maintenant la coder, ainsi que la méthode `retournerLivre`.

Exercice 4 – emprunt et retour d'un livre

Nous codons maintenant les deux dernières méthodes techniques de notre série de TP. Elles permettront d'emprunter et de retourner un livre.

Dupliquez votre code `TP13/ex3` en `TP13/ex4`.

1. Créez, dans le `ControleurLivre`, la méthode

   ```
   public static function emprunterLivre()
   ```

   Si l'adhérent connecté est admin, cette méthode :

   - Récupère le `numLivre` et le `login` en provenance du formulaire précédent,

- Appelle la méthode (pas encore codée)

```
public static function book($n,$l)
```

de la classe `Livre`, qui va entrer la ligne dans la table « emprunte ».

Remarque : pour la `dateEmprunt`, mettez la date n°1, ça suffira, puisqu'en fait on ne tient pas compte de la durée.

- Réaffiche la liste des objets.

Si l'adhérent connecté n'est pas admin, on n'affiche que la liste des objets.

2. Créez la méthode

```
public static function book($n,$l)
```

De la classe `Livre`, qui insère la ligne dans la table emprunte au moyen d'une requête préparée.

3. Créez, dans le `ControleurLivre`, la méthode

```
public static function retournerLivre()
```

Si l'adhérent connecté est admin, cette méthode :

- Récupère le `numLivre` du `$_GET`,
- Appelle la méthode (pas encore codée)

```
public static function giveback($n)
```

de la classe `Livre`, qui va supprimer la ligne de la table « `emprunte` » qui correspond à ce `numLivre`.

- Réaffiche la liste des objets.

Si l'adhérent connecté n'est pas admin, on n'affiche que la liste des objets.

4. Créez la méthode

```
public static function giveback($n)
```

De la classe `Livre`, qui supprime la ligne de la table « `emprunte` » qui correspond à ce `numLivre`, au moyen d'une requête préparée.