

## TP10 – Création et validation de compte par un internaute

Pour terminer les fonctionnalités liés aux sessions et aux comptes, nous allons programmer les deux derniers points importants :

- La création de compte par un internaute non adhérent,
- La validation du compte créé par le biais d'un email de validation.

### Exercice 1 – Adaptation de la page de connexion – un nouveau formulaire

Dupliquez votre TP9/ex7 en TP10/ex1.

Nous arrivons bientôt à la fin de notre démarche. Nous allons maintenant permettre à un internaute non adhérent de créer son compte, et donc de devenir adhérent. Le formulaire de connexion va se transformer pour permettre :

- la connexion pour un adhérent,
- la création de compte pour un non adhérent.

Modifiez le formulaire de connexion pour qu'il intègre une deuxième balise `form` correspondant à la création du compte. L'action associée à ce formulaire sera `creerCompteAdherent`. Ce formulaire proposera tous les champs pour créer un nouvel adhérent. Transférez et testez l'affichage.

The screenshot shows a web browser window with the title "formulaire de connexion". The address bar shows the URL "webdev.iut-orsay.fr/~sgagne/LP/TP/TP9/ex8/index.php". The page contains two side-by-side form boxes. The left box is titled "déjà inscrit ?" and contains fields for "login" and "mot de passe", with a "je me connecte" button below. The right box is titled "pas encore inscrit ?" and contains fields for "login", "mot de passe", "nom", "prénom", and "email", with a "je crée mon compte" button below. The browser's address bar and navigation buttons are visible at the top.

Bibliothèque 2022

*la nouvelle interface de connexion*

## Exercice 2 – ControleurAdherent::creerCompteAdherent et userCreatingAccount

Il va maintenant falloir coder cette méthode `creerCompteAdherent`. Mais il va falloir aussi remodifier l'algorithme du routeur pour que cette nouvelle action soit prise en charge.

Dupliquez votre TP10/ex1 en TP10/ex2.

1. Dans le `controleurAdherent`, créez une méthode

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

qui va :

- récupérer du tableau `$_GET` les `login`, `mdp`, `nomAdherent`, `prenomAdherent` et `email` en provenance du formulaire de création de compte,
- calculer la date d'adhésion (la date courante, sous la forme  
`$d = date("Y-m-d");`
- appeler la méthode `Adherent::addAdherent` avec les paramètres récupérés (la catégorie par défaut sera 1) ;
- réafficher le formulaire de connexion.

2. Dans la classe `Session`, créez sur le modèle des méthodes déjà codées la méthode qui détecte si un internaute est en train de créer son compte :

```
public static function userCreatingAccount() {  
    $bool = isset($_GET["action"]) && $_GET["action"] == "creerCompteAdherent";  
    return $bool;  
}
```

3. Modifiez l'algorithme du routeur pour que, dans le test initial, on ajoute la prise en compte de cette éventuelle création de compte en cours :

```
// si aucun adhérent n'est connecté,  
// et si aucun adhérent n'est en train de se connecter,  
// et si aucun internaute n'est en train de créer son compte,  
// alors :  
//   - l'action est "afficherFormulaireConnexion"  
//   - le contrôleur est "ControleurAdherent"  
// sinon :  
//   - si un contrôleur correct est passé dans l'url, on l'utilise pour $controleur,  
//   - si une action correcte est passée dans l'url, on l'utilise pour $action,  
// enfin, on lance l'action du contrôleur.
```

4. Transférez toutes ces nouveautés. Maintenant, si tout s'est bien passé, vous allez :

- Afficher la page de connexion,
- Créer un nouvel adhérent par le formulaire dédié,
- Vérifier immédiatement sur phpMyadmin que le nouvel adhérent apparaît bien dans la table `Adherent`,
- Connecter ce nouvel adhérent.

Si tout fonctionne, c'est très bien. Sinon, c'est à perfectionner...

### Exercice 3 – Modification de la table `Adherent` pour la validation du compte par email

Habituellement, lorsque vous créez un compte sur un site web « digne de ce nom », vous recevez un email à l'adresse indiquée lors de la création du compte.



Cet email contient un lien qui, une fois cliqué, valide définitivement votre compte. Cela sert à vérifier que votre adresse email est valide.

Tant que le lien n'est pas cliqué, votre compte reste inactif.

Nous allons mettre en place ce mécanisme.

Dupliquez votre TP10/ex2 en TP10/ex3.

1. Commencez par ajouter un champ à la table `Adherent` : appelons ce nouveau champ `chaineValidationEmail`. Donnez lui `NULL` comme valeur par défaut. Ainsi, tous les adhérents déjà présents dans la base auront `NULL` comme valeur pour ce champ.

Nom	Type	Interclassement	Attributs	Null	Valeur par défaut
<b>login</b> 	<code>varchar(50)</code>	<code>utf8_general_ci</code>		Non	<i>Aucun(e)</i>
<b>mdp</b>	<code>varchar(64)</code>	<code>utf8_general_ci</code>		Non	<i>Aucun(e)</i>
<b>nomAdherent</b>	<code>varchar(50)</code>	<code>utf8_general_ci</code>		Oui	<code>NULL</code>
<b>prenomAdherent</b>	<code>varchar(50)</code>	<code>utf8_general_ci</code>		Oui	<code>NULL</code>
<b>email</b>	<code>varchar(50)</code>	<code>utf8_general_ci</code>		Oui	<code>NULL</code>
<b>dateAdhesion</b>	<code>date</code>			Oui	<code>NULL</code>
<b>numCategorie</b> 	<code>int(11)</code>			Non	<i>Aucun(e)</i>
<b>isAdmin</b>	<code>tinyint(4)</code>			Non	<code>0</code>
<b>chaineValidationEmail</b>	<code>varchar(32)</code>	<code>utf8_general_ci</code>		Oui	<code>NULL</code>

2. Modifiez l'action `connecterAdherent` du `controleurAdherent`, pour que la connexion ne puisse se faire que si ce champ est `NULL`. Changez le champ `validationEmail` (mettez une valeur bidon) d'un des adhérents existants pour vérifier que si ce champ n'est pas `NULL`, la connexion est maintenant impossible. Remettez ensuite ce champ à `NULL` (attention, pas la chaîne « `NULL` », mais l'absence de valeur qui va remettre `NULL` par défaut).

#### Exercice 4 – `creerCompteAdherent`, `creerAdherent` et adaptation de la classe `Adherent`

L'ajout de cette chaîne de validation de l'email nous oblige à modifier la classe `Adherent`, ainsi que les méthodes `creerAdherent` et `creerCompteAdherent` du contrôleur, et `addAdherent` du modèle.

Dupliquez votre TP10/ex3 en TP10/ex4.

1. Modifiez l'action `creerCompteAdherent` (réservée à un internaute qui crée son compte). Pour cela, créez une chaîne aléatoire qui servira à remplir le champ `chaineValidationEmail` de la façon suivante :

```
$ch = bin2hex(openssl_random_pseudo_bytes(16));
```

Cette fonction permet de générer des chaînes de 32 caractères aléatoires.

De la même façon, modifiez la méthode `creerCompteAdherent` (qui est réservée à l'internaute qui crée son compte).

2. Modifiez l'action `creerAdherent` (réservée à l'administrateur qui crée en direct un adhérent). On suppose dans ce cas que l'administrateur donne directement la valeur `NULL` à la chaîne de validation, ce qui est logique si on considère par exemple que la création se fait en présence du futur adhérent, avec une validation implicite « en présentiel ».
3. Modifiez le modèle `Adherent` pour que :
  - le nouvel attribut `$chaineValidationEmail` soit pris en compte,
  - la méthode `addAdherent` s'adapte aussi à ce nouvel attribut.
4. A ce stade, transférez, puis créez un nouvel adhérent par le formulaire de création de compte, et vérifiez l'effet dans la base de données. Vous pouvez aussi vous connecter en tant qu'administrateur, créer un nouvel adhérent, et vérifier que de nouveau, il a été créé avec un champ `chaineValidationEmail` pas encore `NULL`.

## Exercice 5 – validateAccount, validerCompteAdherent et userValidatingAccount

Il reste maintenant à coder la phase de validation par le futur adhérent. Attention, exercice long et assez technique...

Dupliquez votre TP10/ex4 en TP10/ex5.

1. Dans cette question, on ajoute la méthode `validateAccount` au modèle `Adherent`.

Cette méthode ressemble un peu à la méthode `checkMDP` :

- elle gère deux paramètres `$l` et `$ch` (`login` et `chaineValidationEmail`),
- elle vérifie s'il existe dans la base un adhérent dont le `login` est `$l` et dont la `chaineValidationEmail` est `$ch`. Si c'est le cas, alors elle passe la `chaineValidationEmail` de cet adhérent à `NULL`.

Codez cette méthode du modèle (il y aura donc deux requêtes dans cette méthode).

2. Codons maintenant l'action `validerCompteAdherent` du `controleurAdherent`. Cette action sera lancée à partir du futur lien cliquable de l'email (l'envoi de l'email sera l'objet de l'exercice 6). Le lien de l'email sera de cette forme :

<.../index.php?controleur=controleurAdherent&action=validerCompteAdherent&login=...&champValidationEmail=...>

Dans l'url de ce lien, il y a toute une partie en pointillés qui donne le début de l'url. Pour cela, vous pouvez utiliser la très pratique constante PHP nommée `__DIR__`, qui indique le répertoire de travail. Lorsqu'on duplique le code, c'est bien plus simple... Il y a aussi le `login` et le champ `chaineValidationEmail`. On va donc, dans la méthode `validerCompteAdherent` :

- récupérer du tableau `$_GET` les `login` et `chaineValidationEmail` (le tableau `$_GET` aura été rempli au moment du clic sur le lien de validation),
- appeler la méthode `validateAccount` du modèle avec ces deux arguments.
- afficher la page avec le formulaire de connexion.

3. Créez la méthode `userValidatingAccount` de la classe `Session`, sur le modèle des autres. Elle servira au routeur à détecter la méthode `validerCompteAdherent`.

```
public static function userValidatingAccount() {  
    $bool = isset($_GET["action"]) && $_GET["action"] == "validerCompteAdherent";  
    return $bool;  
}
```

4. Modifiez l'algorithme du routeur pour que, dans le test initial, on ajoute la prise en compte de cette éventuelle validation de compte en cours :

```
// nouvel algorithme du routeur
// si aucun adhérent n'est connecté,
// et si aucun adhérent n'est en train de se connecter,
// et si aucun internaute n'est en train de créer son compte,
// et si aucun internaute n'est en train de valider son compte,
// alors :
//   - l'action est "afficherFormulaireConnexion"
//   - le contrôleur est "ControleurAdherent"
// sinon :
//   - si un contrôleur correct est passé dans l'url, on l'utilise pour $controleur,
//   - si une action correcte est passée dans l'url, on l'utilise pour $action,
// enfin, on lance l'action du contrôleur.
```

- Enfin, il reste l'envoi du mail qui servira à lancer l'action `validerCompteAdherent`. Dans la méthode `creerCompteAdherent` de `controleurAdherent`, on va écrire cet email et l'envoyer, juste après avoir inséré le nouvel adhérent pas encore validé, et juste avant de réafficher le formulaire de connexion.

Voici le modèle du code qui permet de rédiger un email en mode html et de l'envoyer par la fonction `mail()` de PHP :

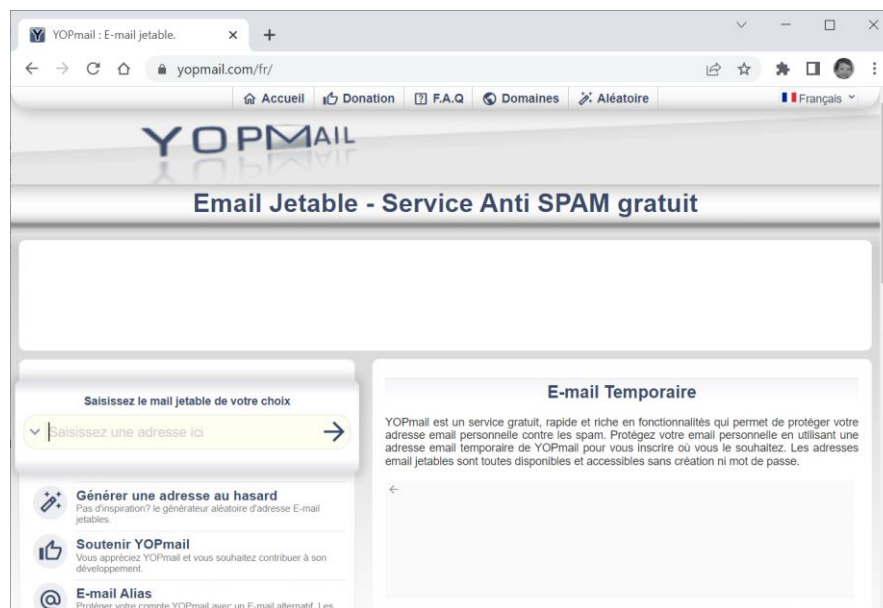
```
// envoi de l'email pour validation du compte
// 1. création du lien
$lien = __DIR__."/index.php?";
$lien .= "controleur=controleurAdherent&action=validerCompteAdherent&login=$l&chaineValidationEmail=$ch";
// 2. Le destinataire de l'email de validation
$destinataire = $_GET["email"];
// 3. Le sujet de l'email
$sujet = "validation de votre compte";
// 4. Le contenu de l'email au format html
$message = "<html><head><meta charset='utf-8'><title>validation de votre compte</title></head><body>";
$message .= "<p>Bonjour $p $n, voici un lien pour valider la création de votre compte associé au login $l : </p>";
$message .= "<p><a style='text-decoration:underline; color:blue;' href='$lien'>valider le compte de $l</a></p>";
$message .= "</body></html>";
// 5. l'entête de l'email
$entete = array(
    'MIME-Version: 1.0',
    'Content-type: text/html; charset=utf-8',
    'From: webmaster@bibliotheque2022.com'
);
// 6. envoi de l'email
mail($destinataire,$sujet,$message,implode("\r\n",$entete));
```

Dans cette capture, `$l` est le login, `$ch` la chaîne de validation, `$n` le nom, `$p` le prénom. Vous adapterez bien sûr l'url de `$lien`.

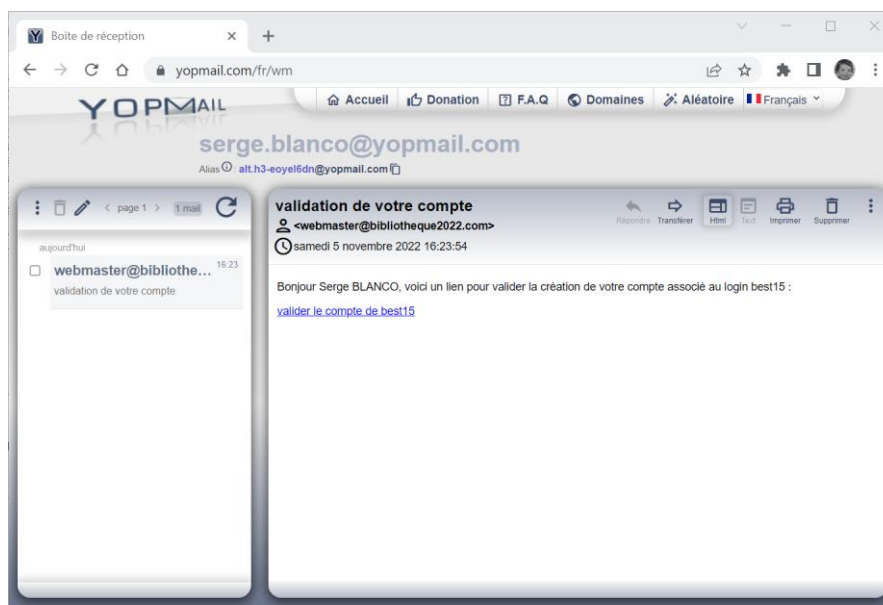
Codez la nouvelle version de `creerCompteAdherent`.

6. Il reste à tester. Nous allons pour cela utiliser des emails « jetables », disponibles sur yopmail.com. Voici le protocole :

- Par le formulaire de la page de connexion, créez un nouvel adhérent. Donnez-lui un email de la forme `nom.prenom@yopmail.com`,
- Vérifiez sur phpMyadmin qu'il a bien été créé, mais que sa chaîne de validation n'est pas encore *NULL*,
- Allez sur la page <https://yopmail.com/fr/>



- Saisissez l'adresse yopmail que vous avez utilisée pour ce nouvel adhérent, et ouvrez sa boîte de messagerie. Vous devriez voir l'email reçu :



- Survolez le lien et vérifiez la cohérence avec ce que vous avez codé,
- Cliquez sur le lien,
- Vérifiez que dans la base, la chaîne de validation a été mise à *NULL*,
- Vérifiez que maintenant, l'adhérent peut se connecter !

## Exercice 6 – vues plus adaptées

Nous allons ajouter deux vues plus parlantes pour cette phase de validation. Il y aura une vue qui oriente vers le lien de validation, et une vue qui avertit que le compte est validé, avec un lien vers le formulaire de connexion.

Dupliquez votre TP10/ex5 en TP10/ex6.

1. Créez la vue `vue/notificationLienValidation.html` qui affiche le message « veuillez consulter votre messagerie pour valider votre compte ! »

Le rendu doit ressembler à :



2. Modifiez la méthode `creerCompteAdherent`. Après l'envoi de l'email de validation, elle ne doit plus renvoyer vers le formulaire de connexion, mais elle doit :
  - inclure `debut.php`,
  - inclure `notificationLienValidation.html`,
  - inclure `fin.html`.

Attention, ne pas inclure de menu bien sûr.



3. Créez la vue `vue/compteValide.html` qui :

- affiche le message « votre compte a bien été validé ! »,
- affiche un lien vers le formulaire de connexion

Le rendu doit ressembler à :



4. Modifiez la méthode `validerCompteAdherent`. Après l'appel de `validateAccount`, elle ne doit plus renvoyer vers le formulaire de connexion, mais elle doit :

- inclure `debut.php`,
- inclure `compteValide.html`,
- inclure `fin.html`.

Attention, ne pas inclure de menu bien sûr.

5. Transférez, créez un nouveau compte par le formulaire et vérifiez que les vues s'enchaînent bien.