TRAVAIL PRATIQUE 3 : INSCRIPTION ET CONNEXION

ÉNONCÉ GÉNÉRAL DU TRAVAIL

Le prochain défi consiste à inscrire vos futurs clients dans la base de données. Ils pourront par après se connecter et, espérons-le, acheter vos produits. La saisie d'informations personnelles sur un site web est un aspect stratégique d'un niveau de complexité relativement élevé, qui implique l'ergonomie et la sécurité. Nous verrons que Symfony offre une panoplie d'outils pour nous aider dans cette tâche.

SPÉCIFICATIONS

1. La page des infos-clients sera accessible par un hyperlien (bouton, image, menu, etc.) situé sur la bannière de votre site.
2. Avant que le visiteur ne soit connecté sur votre site il ne peut que consulter le catalogue et emplir un panier.
3. Avant de pouvoir se connecter, un visiteur doit se créer un compte.
4. Pour créer un compte, le visiteur doit remplir un formulaire par lequel il fournira les informations obligatoires suivantes :
    a. Son nom d'utilisateur, qui peut très bien différer de son nom réel
    b. Son prénom
    c. Son nom de famille
    d. Son genre
    e. Son adresse (incluant numéro civique, rue)
    f. Sa ville
    g. Sa province, sous forme d'une liste déroulante
    h. Son code postal
    i. Un numéro de téléphone
    j. Une adresse courriel
    k. Un mot de passe
    l. Une confirmation du mot de passe
5. Ces informations devront respecter les contraintes suivantes :
    a. Utilisateur : devra être unique dans la BD, longueur minimum de deux caractères, et maximum de 15 caractères;
    b. Prénom : longueur minimum de deux caractères, et maximum de 15 caractères;
    c. Nom de famille : longueur minimum de deux caractères, et maximum de 15 caractères;
    d. Le genre peut prendre trois valeurs : Féminin, Masculin ou neutre
    e. Numéro civique et rue : longueur minimum de deux caractères, et maximum de 15 caractères;
    f. Ville : longueur minimum de deux caractères, et maximum de 15 caractères;
    g. Province : choix de 10 provinces et trois territoires,
    h. Code postal : six caractères, de format A1A1A1, on refuse les lettres (DFOQIU), on accepte les minuscules et majuscules, on peut accepter une espace après les trois premiers caractères;
    i. Numéro de téléphone : dix chiffres. On accepte 10 chiffres collés, ou des parenthèses entourant le code régional, on accepte un espace ou un trait d'union séparant le code régional et/ou séparant les trois chiffres suivants et/ou séparant les quatre derniers chiffres;
    j. L'adresse courriel : doit respecter ce modèle minimal : aaa@aaa.aa
    k. Le mot de passe et la confirmation : ne doivent pas être affichés en clair, ils doivent être identiques, minimum de deux caractères, et maximum de 15 caractères;

    l. Notez que la confirmation du mot de passe **ne doit pas être sauvegardée en BD**, c'est un champ utile seulement à la création du compte et à la modification du mot de passe.

6. Tous les champs sont requis, donc impossible de soumettre le formulaire tant qu'ils ne sont pas tous remplis.
7. Chaque erreur devra être rapportée à l'utilisateur;
8. Un formulaire en erreur sera réinitialisé avec les valeurs fournies précédemment par l'utilisateur excluant le mot de passe;
9. Un formulaire valide dirigera l'utilisateur vers une page de confirmation. Cette page est l'ultime étape avant une tentative d'insertion en BD;
10. Dans la page de confirmation, offrez à l'utilisateur deux options : « annuler » ou « confirmer».
11. Si l'annulation est choisie, on retourne au formulaire d'inscription avec tous les champs pré remplis (sauf le mot de passe); Si l'utilisateur confirme on procède à l'insertion en BD;
12. L'insertion du client en BD devra passer par l'ORM Doctrine;
13. Si lors de l'insertion on viole la contrainte d'unicité du champ utilisateur, on en avise l'utilisateur et on retourne à la page de création d'un compte avec tous les champs pré remplis (sauf le mot de passe);
14. Une fois l'insertion réussie, on avise le client qu'il est connecté et on remplace le lien « créer un compte » par un lien « modifier le compte »
15. On remplace aussi le lien « connexion » par un lien « déconnexion », ainsi on connaît en tout temps le statut de l'utilisateur : connecté ou non;
16. Si l'utilisateur connecté clique sur le lien de modification de son compte on ouvre une page composée de deux formulaires :
    a. Le premier formulaire lui permet de modifier son nom, prénom, ses champs d'adresse, de courriel, de genre et de téléphone. On interdit la modification de son nom d'utilisateur;
    b. Dans le deuxième formulaire on lui permet de modifier son mot de passe;
        i. Pour ce faire il doit fournir son mot de passe actuel, son nouveau mot de passe et sa confirmation. Les trois doivent être validés.
    c. On applique les mêmes validations que lors de la création de son compte sur tous les champs modifiables;
17. Un utilisateur connecté pourra cliquer sur un lien « Commande » lui permettant de transformer un panier en commande. Pour ce TP ce lien est inactif, il sera développé dans le TP4;
18. Un utilisateur non-connecté pourra se connecter en cliquant le lien de connexion. Ce lien ouvre un écran de connexion qui saisit le nom de l'utilisateur et son mot de passe et effectue la validation;
19. Si la connexion est valide on adapte le site selon le mode client-connecté.
20. Une création de compte entraîne un message éclair (flash) d'accueil, de même qu'une connexion réussie.
21. Une modification d'information entraîne un message éclair de confirmation
22. Une déconnexion entraîne un message éclair d'au revoir.

ÉTAPES DE DÉVELOPPEMENT

1- Créez-une entité Symfony nommée Client, manuellement ou avec la console :
`php bin/console make:entity`
2- Cette entité sera composée des champs décrits à la spécification 4 (sauf le champ confirmation);
3- Le champ « utilisateur » sera unique;
4- Aucun des champs ne sera « nullable », ils sont donc tous obligatoires;
5- Une fois l'entité Client bien générée, vous pouvez créer la table correspondante dans votre BD. Encore une fois vous avez deux choix : manuellement ou avec maj.bat, qui exécutera un:
`php bin/console doctrine:schema:update --force`

6- Pour ce TP, les mots de passe seront stockés en clair dans la BD. Aucun hachage pour l'instant .

7- Créez-un ClientType (form builder de Symfony) manuellement ou avec la console :
`php bin/console make:form`

8- Créez un nouveau « controller » pour les trois pages client.
   a. Création du compte
   b. Confirmation des informations
   c. Modifications des informations

9- Les pages Création et Modification utiliseront la même « form » ClientType; cependant cette form ne sera pas présentée de la même façon dans les deux contextes, vous devrez adapter ces présentations. De plus la page Modification devra ajouter un formulaire pour la modification du mot de passe.

10- Commencez par la page de création et vérifier que vous pouvez « renderer » le formulaire.
```
$client = new Client();
$formClient = $this->createForm(ClientType::Class, $client);
$this->render('CreationClient.html.twig', array('formClient'=>
$formClient->CreateView())
```

11- Dans le fichier CreationClient.html.twig vérifiez rapidement que le formulaire s'affiche avec la fonction :
```
{{form(formClient)}}
```

12- Si tout se passe bien décortiquez plus finement l'affichage avec un bloc pour chaque champ. Par exemple :
```
…
<div>
{{ form_label(formClient.utilisateur, "Utilisateur") }}
{{ form_widget(formClient.utilisateur) }}
{{ form_errors(formClient.utilisateur) }}
<div>
…
```

13- Si ce n'est déjà fait, ajoutez un bouton « submit » au formulaire (ClientType.php), ce submit rappellera la méthode du contrôleur qui gère l'affichage de la page d'inscription

14- Ce contrôleur devra gérer les deux situations suivantes :
   a. Affichage initial du formulaire
   b. Validation des infos du formulaire, donc détection de la soumission par la méthode **`$form->isSubmitted()`**

15- Si vous êtes dans la situation de validation des valeurs des champs, le formulaire devra appeler les méthodes
```
$formClient->handleRequest($request);
```
et
```
$formClient->isValid();
```

16- En cas d'erreurs le formulaire sera de nouveau « renderé » mais cette fois en affichant les erreurs à l'utilisateur.

17- Les validations à appliquer sont définies à la spécification 5-. Elles seront spécifiées dans l'entité Client.php. Pour spécifier les validations ajoutez-y le «use» suivant
```
use Symfony\Component\Validator\Constraints as Assert;
```
Et dans l'annotation de chaque champ utilisez le « Assert » approprié : Par exemple :

```
. . .
    #[ORM\Column()]
    #[Assert\Length(min:2, minMessage:'deux caractères minimum')]
    #[Assert\Length(max:15, maxMessage:'cent caractères maximum')]
    private ?string $nom = null;
. . .
```

18- En plus des validations métiers il faut aussi s'assurer de l'unicité du champ utilisateur, en effet chaque nom d'utilisateur doit être unique. On peut automatiser cette vérification ainsi dans l'entité Client.php :

```
. . .
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
. . .
#[ORM\Entity(repositoryClass: ClientRepository::class)]
#[UniqueEntity(fields:"utilisateur", message:"existe déjà en BD")]
class Client
{
. . .
```

19- Si aucune erreur ne subsiste, vérifiez la concordance entre le mot de passe désiré et sa confirmation;

20- Si tout est valide. Conservez l'entité Client dans une variable de session « client_candidat »
`$request->getSession()->set('client_candidat', $client);`
et redirigez l'utilisateur vers une page de confirmation
`$this->redirectToRoute(…).`

21- Cette page affichera les infos en « read only » et n'offrira que deux choix à l'utilisateur : Confirmation ou Annulation.

22- L'annulation ramènera l'utilisateur à la page d'inscription, la confirmation provoquera une tentative d'insertion en BD. Pour ce faire, utilisez l'ORM de Doctrine. Par exemple : :

```
$client = $request->getSession()->get('client_candidat');
$em = $doctrine->getManager();
$em->persist($client);
$em->flush();
```

23- À ce stade l'utilisateur est considéré comme un client connecté, vous pouvez enlever la variable de session client_candidat et ajouter la variable de session « client_connecte » :

```
…
$request->getSession()->remove('client_candidat');
$request->getSession()->set('client_connecte',
                            $client->getUtilisateur());
…
```

24- Un client connecté peut modifier ses informations ou se déconnecter. Donc, permutez les liens « Créer un compte » pour « Modifier un compte » et « Connexion » pour « Déconnexion »

25- La modification d'un compte utilisera le même contrôleur, la même entité, le même ClientType (formulaire) ainsi que la même vue (twig) que la création d'un compte. La vue aura donc besoin de distinguer deux contextes différents (création vs modification). Voir à ce sujet les spécifications 15- et suivantes.

26- Vous aurez besoin d'utiliser la commande twig « do » pour signifier à Symfony de ne pas afficher certains champs (utilisateur, mot de passe et confirmation) dans le contexte Modification:

```
...
{% do formClient.utilisateur.setRendered %}
…
```

Attention : si vous avez généré votre entité Client avec la console, vérifier le typehint du setUtilisateur() (et des autres champs setRendered) si celui-ci est **string** il faudrait le modifier pour **?string**. Ceci est nécessaire parce que le {% do formClient.nomDuChamp.setRendered %} enverra une valeur null pour ce champ et le HandleRequest() utilisera le setnomDuChamp() qui refusera une valeur null, à moins que le typehint soit **?string** (null ou string) , ou qu'il n'y ait pas de typeHint.

Il vous reste à

a. Gérer la mise à jour en BD des infos client par Doctrine

b. Définir un écran de connexion qui permettra à un utilisateur non connecté de se connecter
c. Définir un lien de déconnexion qui fera un « remove » de la variable de session et permutera les liens vers le contexte non-connecté
d. Peaufiner le rendu visuel de vos formulaires en utilisant les CSS de votre thème.
e. Vous assurez que la page de modification est protégée contre le piratage par injection URL
f. Gérer les « FlashBags » pour la rétroaction sur chaque changement d'état de votre site

EXIGENCES DE REMISE

1. Votre site sera codé avec le framework Symfony, version 7
2. Vous avez trois niveaux de remise :
   - **Léa**, fichiers sources uniquement (les dossiers **public, src** et **templates**)
   - **Production** : Votre site sera publié sur le serveur web du département (techinfo-cstj.ca) dans un sous-domaine de votre domaine de base, nommé tp3. Par exemple: https://tp3.1234567.techinfo-cstj.ca/
     Ajustez l'index.php de votre racine personnelle pour qu'elle offre un hyper lien vers votre tp3
   - **Github.com**, dans votre remise Léa ajoutez un fichier gitHub_URL.txt, indiquant l'URL de votre repository pour ce projet sur GitHub, mettez AlainProf comme collaborateur de votre repository.
     Sur git, après votre commit/push final dans la branche master, créez une « branch » tp3 et « commitez » + push avec cette branche. C'est cette branche que j'utiliserai pour vous corriger.
3. Votre site sur techinfo doit être configuré « prod »
4. Les requêtes à la BD utiliseront l'ORM Doctrine.
5. Vous devez utiliser des variables de session pour gérer les cycles de vie de vos clients.
6. L'accès à la BD de production passera par un utilisateur BD spécifiquement créé pour ce projet
7. Votre site devra être actif en tout temps et **ne devra pas être modifié entre la date de remise et le moment où les notes seront publiées**
8. Tous vos fichiers sources devront être bien **identifiés** et **commentés**.
9. Le HTML généré par votre code devra être conforme au **HTML5**.
10. Ce travail doit être réalisé individuellement.
11. ***Remise: 11 avril 2025 avant 23 h 59.***

ÉVALUATION

Pondération: 16% de la session.

GRILLE DE CORRECTION

Hyperlien pour accéder à la création d'un compte	2
Hyperlien pour accéder à la modification d'un compte (bascule de la création d'un compte)	3
Hyperlien et fonctionnalité pour la connexion	1

Hyperlien et fonctionnalité pour la déconnexion (bascule du lien connexion)	3
Formulaire d'inscription avec tous les champs présents	13
Formulaire de modification (utilisateur non modifiable et sans mot de passe et confirmation)	10
Page de modification protégée contre injection URL	2
Formulaire de modification du mot de passe avec trois champs	7
Formulaire de connexion	5
Validation de toute les informations	13
Unicité du nom d'utilisateur	5
Rétroaction ergonomique sur les erreurs dans chacun des contextes	10
Rétroaction générale lors de la création, modification, connexion et déconnexion	9
Écran de validation finale de l'inscription et connexion automatique lors de la confirmation de la création d'un compte	6
Pérennisation de la connexion par variable de session	7
Vidage appropriée de la variable de session lors de la déconnexion	4
Aucune validation cliente (HTML) : Toutes les validations sont faites au niveau serveur	-20
Uniformité de l'affichage des pages (bannière, pied de page, thème, etc.)	-15
Respect des exigences	-30
Faute de français	-10
Total	100