TP13 – personnalisation

Introduction, Contexte et Objectifs

Pour le moment, nous avons différents types d'utilisateurs :

- le superuser, qui a un accès privilégié à la page par défaut d'administration proposée par django,
- les utilisateurs staff, à qui on a donné les droits classiques du CRUD (create, read, update, delete)
- les clients connectés,
- les internautes non connectés.

Tout ceci se fait grâce au **modèle User par défaut de Django**, de sorte qu'on n'a pas eu à coder de modèle pour l'application applicompte. Mais ce modèle va s'avérer incomplet, car nous allons ajouter une image au profil d'un utilisateur. Nous allons donc définir un modèle User personnalisé **: PizzaUser**

Il nous restera aussi dans ce TP à adapter toutes les views (passage de User à PizzaUser), et à créer la page de profil, puisque c'est du grand classique.

Personnalisation du modèle User

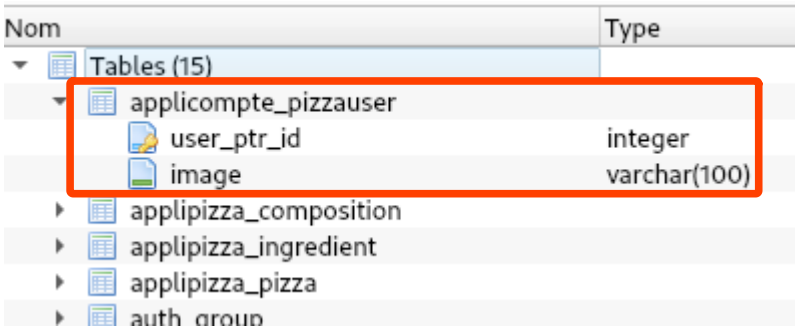
1. Dans le fichier models.py d'applicompte, créez un modèle PizzaUser qui étend User (le modèle django User vient de django.contrib.auth.models), avec un attribut supplémentaire nommé image. Comme cet attribut servira à modéliser la photo de l'utilisateur, il faut, comme pour les images de pizzas, prévoir un fichier par défaut et un répertoire d'upload :

webpizza > applicompte > models.py > PizzaUser

```python
from django.db import models
from django.contrib.auth.models import User

# Create your models here.
class PizzaUser(User) :
    # fichier image de l'utilisateur
    image = models.ImageField(default = 'imagesUsers/default.PNG', upload_to = 'imagesUsers/')
```

2. Mettez en place la migration de ce nouveau modèle. Vérifiez que, dans la base de données, vous avez maintenant une nouvelle table applicompte_pizzauser :



Concrètement vous avez un champ qui permet de faire la jointure avec la table user, et le champ supplémentaire image, défini dans le modèle PizzaUser.

3. Faites en sorte que le modèle PizzaUser puisse être géré par le superuser via l'interface d'administration proposée par django (comme cela a été fait pour Pizza, Ingredient et Composition).

4. Connectez-vous en tant qu'administrateur sur le portail de django. Créez un nouvel utilisateur (pas en ajoutant un nouvel User d'applicompte, mais un nouvel User classique). Ensuite, dans sqlitebrowser, vérifiez que le nouvel

utilisateur existe bien, et complétez à la main le champ « image » correspondant dans la table applicompte_pizzauser. Faites-le pour tous les utilisateurs existants. Cela peut donner un résultat comme :

Table : auth_user

	id	password	last_login	is_superuser	username	last_name	email	is_staff	is_active
	F...	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	pbkdf2_sha...	2023-07-17 ...	1	sgagne		sgagne@email.com	1	1
2	2	pbkdf2_sha...	2023-07-16 ...	0	gaetano	scirea	gaetano@email.com	1	1
3	3	pbkdf2_sha...	2023-07-17 ...	0	claudio	gentile	claudio@email.com	0	1
4	7	pbkdf2_sha...	NULL	0	marco	tardelli	marco@email.com	0	1

Table : applicompte_pizzauser

	user_ptr_id	image
	Filtre	Filtre
1	1	imagesUsers/default.PNG
2	2	imagesUsers/default.PNG
3	3	imagesUsers/claudio.PNG
4	7	imagesUsers/marco.PNG

Vérifiez que vous pouvez vous connecter et vous déconnecter avec ce nouvel utilisateur.

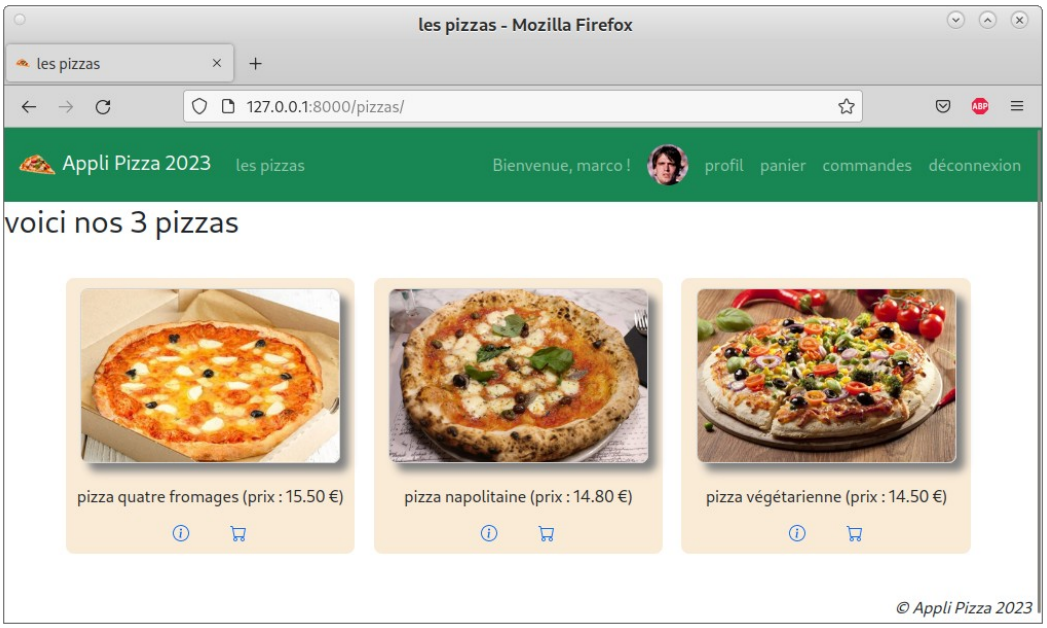
5. Récupérez localement les photos des futurs utilisateurs proposées sur Moodle (ou choisissez-en d'autres si vous préférez). Elles serviront aux futurs clients.

   Créez un répertoire images/imagesUsers (en parallèle de imagesPizzas) et placez-y les photos utiles.

   Nous allons maintenant personnaliser la page de l'utilisateur connecté.

Personnalisation du menu

Nous allons faire en sorte que maintenant, quand un utilisateur est connecté, il ait sa photo affichée, avec un message personnalisé, et un lien vers sa page de profil, en plus des liens panier et commandes :



1. Adaptez d'abord, dans toutes les views (y compris connexion), l'utilisateur récupéré. Voici par exemple la view pizzas modifiée :

```python
def pizzas(request) :
    user = None
    if request.user.is_authenticated :
        user = PizzaUser.objects.get(id = request.user.id)

    # récupération des pizzas de la base de données
    # avec les mêmes instructions que dans le shell
    lesPizzas = Pizza.objects.all()

    # on retourne l'emplacement du template et, même
    # s'il ne sert pas cette fois, le paramètre
    # request, ainsi que le contenu calculé (lesPizzas)
    # sous forme d'un dictionnaire python
    return render(
        request,
        'applipizza/pizzas.html',
        {'pizzas' : lesPizzas, "user" : user}
    )
```

Le passage, dans toutes les vues, de User à PizzaUser est indispensable pour récupérer la photo.

2. Adaptez ensuite les trois menus pour qu'ils affichent la photo avec message adapté (pour client et staff) ou juste un message de bienvenue (pour utilisateur non connecté) :

```
<ul class="navbar-nav">
    <li class="nav-item"><span class="nav-link">Bienvenue, {{user}} ! </span></li>
    <li class="nav-item"><span class="nav-link"><img src="{{MEDIA_URL}}{{user.image}}" class="rounded-circle" width="40" height="40" alt="Avatar" /></span></li>
    <li class="nav-item"><a class="nav-link" href="/user/update/"> profil </a></li>
    <li class="nav-item"><a class="nav-link" href="/cart/"> panier </a></li>
    <li class="nav-item"><a class="nav-link" href="/orders/"> commandes </a></li>
    <li class="nav-item"><a class="nav-link" href="/logout/"> déconnexion </a></li>
</ul>
```

```
<ul class="navbar-nav">
    <li class="nav-item"><span class="nav-link">Bienvenue, {{user}} ! </span></li>
    <li class="nav-item"><span class="nav-link"><img src="{{MEDIA_URL}}{{user.image}}" class="rounded-circle" width="40" height="40" alt="Avatar" /></span></li>
    <li class="nav-item"><a class="nav-link" href="/logout/"> déconnexion </a></li>
</ul>
```

```
<ul class="navbar-nav">
    <li class="nav-item"><span class="nav-link">Bienvenue ! </span></li>
    <li class="nav-item"><a class="nav-link" href="/register/"> inscription </a></li>
    <li class="nav-item"><a class="nav-link" href="/login/"> connexion </a></li>
</ul>
```

Pour le moment, le lien profil (comme les liens panier et commandes) ne sont pas opérationnels.

La page de profil

Concrètement, c'est une page de modification du PizzaUser (sauf le mot de passe qui passera par une procédure spéciale, voir TP suivant).

Dans la capture du menu « client connecté » précédente, on voit un lien de la forme

```
<a class="nav-link" href="/user/update/"> profil </a>
```

1. Dans le fichier urls.py d'applicompte, ajoutez le path

```
path('user/update/', views.formulaireProfil),
```

2. Créez ensuite dans views.py d'applicompte la view suivante :

```python
def formulaireProfil(request) :
    # création du user
    user = None
    # cas d'un utilisateur connecté
    if request.user.is_authenticated :
        user = PizzaUser.objects.get(id = request.user.id)
        return render(
            request,
            'applicompte/profil.html',
            {"user" : user}
        )
    # cas d'un internaute non connecté
    else :
        return render(
            request,
            'applicompte/login.html',
        )
```

3. Créez un fichier applicompte/forms.py avec une classe PizzaUserForm comme cela avait été déjà fait dans applipizza :

webpizza > applicompte > forms.py > ...

```python
from django.forms import ModelForm
from applicompte.models import PizzaUser

class PizzaUserForm(ModelForm) :
    class Meta :
        model = PizzaUser
        fields = ['username','first_name','last_name','email','image']
```

4. Créez ensuite le template applicompte/profil.html

webpizza > applicompte > templates > applicompte > profil.html > ...

```
{% extends 'applipizza/base.html' %}
{% block title %} profil {% endblock %}
{% block main %}
<h2>votre profil</h2>
<form enctype="multipart/form-data" id="form_profil" method="post" action="/user/{{user.id}}/updated/">
    {% csrf_token %}
    <div class="mb-3">
      <label for="id_username"> identifiant </label>
      <input type="text" class="form-control" id="id_username" name="username" value="{{user.username}}">
    </div>
    <div class="mb-3">
        <label for="first_name"> prénom </label>
        <input type="text" class="form-control" id="first_name" name="first_name" value="{{user.first_name}}">
    </div>
    <div class="mb-3">
        <label for="last_name"> nom </label>
        <input type="text" class="form-control" id="last_name" name="last_name" value="{{user.last_name}}">
    </div>
    <div class="mb-3">
        <label for="email"> email </label>
        <input type="email" class="form-control" id="email" name="email" value="{{user.email}}">
    </div>
    <div class="mb-3">
        <div style="display:flex;flex-direction: column;">
            <label for="id_image" class="form-label">votre photo</label>
            <img class="imageUser" src="{{MEDIA_URL}}{{user.image}}" alt="">
        </div>
        <input class="form-control" name="image" type="file" accept="image/*" id="id_image" value="{{user.image}}">
    </div>
    <button type="submit" class="btn btn-primary">mettre à jour</button>
  </form>
{% endblock %}
```

Ce formulaire affiche les valeurs actuelles de l'identifiant, des prénom, nom, email, et l'image actuelle. Testez l'accès à ce formulaire.

Ce formulaire évoque une action="/user/updated/". Il faut donc prévoir cette action dans views.py et le path correspondant dans urls.py.

5. Dans urls.py, ajoutez le path

path('user/updated/', views.traitementFormulaireProfil),

6. Dans views.py, écrivez la view de traitement suivante, en comprenant bien les étapes :

- récupération du PizzaUser
- récupération du formulaire (instance : ce PizzaUser)
- sauvegarde du PizzaUser via la sauvegarde du formulaire
- récupération du PizzaUser modifié
- récupération des pizzas
- appel de pizzas.html

```python
def traitementFormulaireProfil(request) :
    user = None
    # cas d'un utilisateur staff
    if request.user.is_authenticated :
        user = PizzaUser.objects.get(id = request.user.id)
        # user = PizzaUser.objects.get(id = user_id)
        form = PizzaUserForm(request.POST, request.FILES, instance = user)
        if form.is_valid() :
            form.save()
            user = PizzaUser.objects.get(id = request.user.id)
        lesPizzas = Pizza.objects.all()
        return render(
            request,
            'applipizza/pizzas.html',
            {"pizzas" : lesPizzas, "user" : user}
        )
    else :
        return render(
            request,
            'applicompte/login.html',
        )
```

7. Vérifiez que tout fonctionne, y compris la mise à jour d'image, et que cela s'actualise sur la barre de menu. Sinon, cherchez l'erreur...