

Programmation en Python Master 2 Réseaux Télécoms TP 12

Dans ce TP, on reprend le code réalisé lors du TP 11.

1 Éviter les appels des fonctions à partir d'un gabarit HTML

Regardons de près l'instruction de boucle suivante dans le gabarit home.html:

{% for tool in user.tool_set.all %}

Ici, l'expression user.tool_set.all correspond à la valeur retournée par la fonction suivante :

user.tool_set.all()

où l'objet user correspond à l'utilisateur authentifié, et l'attribut tool_set est le soit-disant gestionnaire de liaison. Ce dernier est ajouté automatiquement par le système ORM de Django à chaque modèle référencé par une clé étrangère. Le gestionnaire de liaison permet d'inverser la relation *Many-to-One* matérialisée par la clé étrangère, en la transformant en relation *One-to-Many*. Le nom de ce gestionnaire est formé à partir du nom du modèle en minuscules, en y ajoutant la terminaison _set. Django offre d'autre types de gestionnaires, par exemple, on a déjà utilisé le gestionnaire objets dans l'expression

Tool.objects.filter(id__in=to_delete).delete()

dans la view delete_tools.

Normalement, les appels des fonctions à partir d'un gabarit HTML doivent être évités (de toute façon, ils sont limités aux fonctions sans arguments). La bonne pratique est de ne passer dans le gabarit que des valeurs pré-calculées dans une view. Le méthodes de calcul sont souvent placées dans des fichiers source séparés, pour ne pas encombrer views.py.

Créez un fichier utility.py dans le répertoire coop, avec le contenu suivant :

```
def get_owned_tools(user):
    return user.tool_set.all()
```

Importez cette fonction dans le fichier views.py:

```
from .utility import get_owned_tools
```

puis modifiez le code de la fonction home dans ce fichier pour ajouter dans le dictionnaire de contexte la liste des outils appartenant à l'utilisateur :

```
if request.user.is_authenticated:
    context['owned'] = get_owned_tools(request.user)
```

Finalement, on va remplacer l'appel d'une fonction dans le gabarit HTML par la valeur pré-calculée :

```
{% for tool in owned %}
```

Vérifiez que l'application fonctionne correctement.

2 Modèle de prêts

Les prêts des outils seront représentes par des objets persistants correspondants au modèle suivant :

```
class Lease(models.Model):
    lessee = models.ForeignKey(User, on_delete=models.CASCADE)
    thing = models.ForeignKey(Tool, on_delete=models.CASCADE)
    start = models.DateTimeField(auto_now_add=True)
    stop = models.DateTimeField(null = True)
```

Ici les attributs lessee et thing correspondent à la personne empruntant l'outil et à l'outil emprunté. Les champs start et stop représentent les temps du début et de la fin du prêt. Notez que le champ start est créé avec l'argument auto_now_add=True, pour estampiller chaque nouveau prêt automatiquement avec le temps de l'horloge du SGBD. Notez aussi que le temps de la fin du prêt est nul par défaut; il restera nul tant que le prêt ne sera pas terminé.

Ajoutez ce modèle dans le fichier models.py et effectuez les migrations pour matérialiser le modèle dans la base de données.

3 Liste des outils disponibles

Un outil est disponible lorsqu'il n'existe pas de prêt ouvert qui le concerne. La liste des outils disponibles est donc retournée par la fonction suivante :

```
def get_available_tools():
       return Tool.objects.filter(~Exists(
          Lease.objects.filter(thing=OuterRef('pk'),
                             stop__isnull=True)))
Ajoutez cette fonction dans le fichier utility.py, avec les instruction d'importa-
tion suivantes:
   from django.db.models.expressions import OuterRef
   from .models import Tool, Lease
   from django.db.models import Exists
Ajoutez l'appel de cette fonction dans la view home :
       context['available'] = get_available_tools()
Ajoutez également le code suivant dans le gabarit home.html:
<h2>Vous pouvez emprunter les outils suivants:</h2>
   <form action="" method="post">
       {% csrf_token %}
       <thead>
       <t.r>
       Id
       Description
       <input type="submit" value="Louer" />
       </thead>
       {% for tool in available %}
              {{tool.id}}
              \t{td}{\t<d}
              <input type="checkbox" value="{{tool.id}}"</pre>
                 name="to_borrow" />
              {% endfor %}
```

Placez ce code après les commandes d'affichage du formulaire d'ajout d'un outil, voir Figure 1. Notez que l'URL de l'action est vide pour l'instant, et qu'on utilise la liste available passée dans le contexte dans la *view* home.

Vérifiez que la liste des outils disponibles s'affiche correctement.

</form>

4 Emprunts

```
Pour gérer les emprunts, on ajoutera la fonction suivante dans le fichier views.
ру:
def borrow_tools(request):
    to_borrow=[int(s) for s in request.POST.getlist('to_borrow')]
    for tid in to_borrow:
       l=Lease(lessee=request.user, thing=Tool.objects.get(pk=tid))
       1.save()
    return redirect('home')
Notez que chaque emprunt correspond à un nouvel objet de type Lease ajouté
dans la base de données. Ajouter l'URL correspondant à cette view dans le fichier
urls.py:
urlpatterns = [
   path('', views.home, name = "home"),
   path('add-tool/', views.add_tool, name = "add-tool"),
   path('delete-tools/', views.delete_tools, name = "delete-tools"),
   path('borrow-tools/', views.borrow_tools, name = "borrow-tools"),
]
et modifiez l'action de la forme correspondante dans le gabarit home.html:
        <form action="{% url 'borrow-tools' %}" method="post">
```

Vérifiez qu'un outil emprunté disparait de la liste des outils disponibles.

5 Restitutions

La restitution des outils est gérée par la fonction suivante, qu'on ajoutera dans le fichier views.py :

```
def restitute_tools(request):
    to_restitute=[int(s) for s in request.POST.getlist('to_resitute')]
    Lease.objects.filter(thing__in=to_restitute).update(stop=Now())
    return redirect('home')
```

Pour utiliser la fonction Now(), on ajoutera cette commande d'importation :

```
from django.db.models.functions import Now
```

Notez que dans la fonction restitute_tools on met à jour les prêts concernant les outils dont les identifiants sont présents dans la liste to_restitute, en spécifiant le temps de la fin du prêt.

Ajouter l'URL correspondant à cette fonction dans le fichier urls.py:

```
urlpatterns = [
   path('', views.home, name = "home"),
   path('add-tool/', views.add_tool, name = "add-tool"),
   path('delete-tools/', views.delete_tools, name = "delete-tools"),
   path('borrow-tools/', views.borrow_tools, name = "borrow-tools"),
   path('restitute-tools/', views.restitute_tools,
        name = "restitute-tools"),
]
Pour former la liste des outils qu'on peut restituer, on utilisera la fonction suivante
(à mettre dans utility.py):
def get_borrowed_tools(user):
   return Tool.objects.filter(Exists(
       Lease.objects.filter(thing=OuterRef('pk'),
                           lessee=user, stop__isnull=True)))
On ajoutera la valeur retournée par cette fonction dans le contexte passé pour le
gabarit HTML:
context['borrowed'] = get_borrowed_tools(request.user)
Ajoutez également le code suivant dans le gabarit home.html, de tel sorte que la
nouvelle table s'affiche vers la fin de la page (voir Figure 1):
<h2>Vous empruntez en ce moment les outils suivants:</h2>
   <form action="{% url 'restitute-tools' %}" method="post">
       {% csrf_token %}
       <thead>
              \langle t.r \rangle
              Id
              Description
              <input type="submit" value="Rendre" />
              </thead>
              {% for tool in borrowed %}
                      {td>{{tool.id}}
                      {{tool}}
                      <input type="checkbox" value="{{tool.id}}"</pre>
                          name="to_resitute" />
```

```
{% endfor %}

</form>
```

Vérifiez que les outils empruntés apparaissent dans cette nouvelle table, et qu'il redeviennent disponibles après restitution.

6 Attributs dynamiques

La ligne suivante

```
thing = models.ForeignKey(Tool, on_delete=models.CASCADE)
```

dans le code de la classe Lease signifie que l'enlèvement d'un outil par son propriétaire efface de la base de données tous les prêts concernant cet outil, y compris les prêts en cours. Ce n'est pas forcement le comportement souhaité, il est plus logique de ne pas autoriser à un propriétaire de retirer un outil concerné par un prêt en cours. On peut le faire à l'aide d'un mécanisme d'attributs dynamiques de Django. Modifiez le code de la fonction get_available_tools dans le fichier utility.py de la façon suivante :

Notez que l'effet de ce code est l'ajout d'un attribut dynamique (une annotation) à l'objet Tool. Cet attribut booléen nommé leased correspond au fait que l'outil est concerné par un prêt en cours. On peut maintenant utiliser cet attribut pour enlever les cases à cocher dans la colonne "Supprimer". À cette fin, entourez l'instruction correspondante dans home.html par des commandes conditionnelles :

```
{% if not tool.leased %}
      <input type="checkbox" value="{{tool.id}}" name="to_delete" />
{% endif %}
```

Vérifiez qu'on ne peut plus retirer les outils concernés par les prêts en cours.

Page personnelle de Jean Dupont

Logout

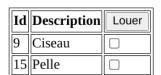
Vous possedez les outils suivants:

Id	Desciption	Prix	Supprimer
8	Tournevis	5.00	
9	Ciseau	20.00	
13	Pompe	80.00	

Ajouter un outil:



Vous pouvez emprunter les outils suivants:



Vous empruntez en ce moment les outils suivants:

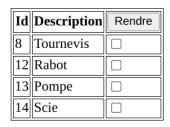


FIGURE 1 – L'interface finale de l'application.