Class-based Views

https://github.com/chezmarcbrown/classdemo-2022-11-17-classbasedviews.git

> Main listing page

urls.py

```
   path("", views.ListingListView.as_view(), name="index"),
```

views.py

```
class ListingListView(ListView):
   model = Listing
   template_name = "auctions/index.html"
   context_object_name = "listings"

   def get_queryset(self):
       return Listing.objects.all().filter(active=True).order_by('-created_at')

   def get_context_data(self):
       context = super().get_context_data()
       context['banner'] = 'Active Listings'
       return context
```

> Create a new listing

urls.py

```
   path("create_listing", views.ListingCreateView.as_view(), name="create-listing"),
```

views.py

```
class ListingCreateView(LoginRequiredMixin, CreateView):
   model = Listing
   form_class = ListingForm
   template_name = "auctions/new_listing.html"
   #fields = ('title', 'description', 'starting_bid', 'categories', 'image')

   success_url = reverse_lazy("index")

   def form_valid(self, form):
       form.instance.creator = self.request.user
       return super().form_valid(form)
```

> Boring so far… but let’s add category pages:

urls.py

```
    path("categories", views.CategoryListView.as_view(), name="categories"),
    path("category/<int:category_id>", views.CategoryListingsView.as_view(), name="category"),
```

views.py

```
class CategoryListView(ListView):
    model = Category
    template_name = "auctions/categories.html"
    context_object_name = "categories"

class CategoryListingsView(ListView):
    model = Listing
    template_name = "auctions/index.html"
    context_object_name = "listings"

    def get_queryset(self):
        return Listing.objects.filter(categories=self.kwargs["category_id"], active=True)

    def get_context_data(self):
        context = super().get_context_data()
        category_id = self.kwargs["category_id"]
        category = Category.objects.get(id=category_id)
        context['category'] = category
        context['banner'] = f'{category.name} Category ({self.get_queryset().count()} listings)'
        return context
```

> Let the fun begin – we will give UIs to handle categories:

1) Add a new category
2) Delete an existing category
3) Change name and/or image of a category

> Add new category

In categories.html, add:

```
<button onclick="location.href='{% url "create-category" %}'">
   New Category
</button>
```

In urls.py, add:

```
   path("create_category", views.CategoryCreateView.as_view(), name="create-category"),
```

In views.py, implement:

```
class CategoryCreateView(CreateView):
   model = Category
   fields = ("name", "image")
   success_url = reverse_lazy("categories")

   def get_context_data(self, **kwargs):
       context = super().get_context_data(**kwargs)
       context["title"] = "Add a new category"
       return context
```

New template file: category_form.html:

```
{% extends "auctions/layout.html" %}
{% load static %}

{% block main %}

<h3>{{ title }}</h3>
<form method="post" enctype="multipart/form-data">
   {% csrf_token %}
   <table>
       {{ form.as_table }}
   </table>
   <input type="submit" value="Submit">
   <input value="Cancel" type="button" onclick="location.href='{% url "categories" %}'">
</form>

{% endblock %}
```

>Delete Category

Rather than use index.html template, use a copy - category_listings.html. This will allow us to put buttons for DELETE (and later EDIT).

Catagory_listings.html:

```
{% extends "auctions/layout.html" %}
{% load static %}

{% block main %}

<h2>{{banner}}</h2>
```

```
<button onclick="location.href='{% url "delete-category" category.id %}'">
   Delete Category
</button>
<button onclick="location.href='{% url "update-category" category.id %}'">
   Edit Category
</button>

{% include "auctions/listings.html" %}

{% endblock %}
```

Views.py - use this template:

```
class CategoryListingsView(ListView):
   model = Listing
   template_name = "auctions/category_listings.html"
   context_object_name = "listings"
```

Add Delete urls:

```
   path("category/delete/<int:pk>", views.CategoryDeleteView.as_view(), name="delete-category"),
```

Add code for the view (template name defaults to the one given):

```
class CategoryDeleteView(DeleteView):
   model = Category
   success_url = reverse_lazy("categories")
   template_name = "auctions/category_confirm_delete.html"
```

Need a confirm delete template category_confirm_delete.html:

```
{% extends "auctions/layout.html" %}
{% block main %}

<form method="post">{% csrf_token %}
```

```
    <p>Are you sure you want to delete category "{{ object }}"?</p>
    {{ form }}
    <input type="submit" value="Confirm">
</form>

{% endblock %}
```

>Update Category (no html, no form)

Add button:

```
<button onclick="location.href='{% url "update-category" category.id %}'">
    Edit Category
</button>
```

Add view:

```
class CategoryUpdateView(UpdateView):
    model = Category
    fields = ("name", "image")
    success_url = reverse_lazy("categories")
```

Add url:

```
    path("category/update/<int:pk>", views.CategoryUpdateView.as_view(), name="update-category"),
```
