Exercise 12: Creating a ListView

20 to 30 minutes

In this exercise, you will create the page for listing jokes. To do this, you will inherit from `ListView`.

1. Open `jokes/views.py` in your editor and delete the current contents.
2. Import `ListView` from `django.views.generic`:

   ```
   from django.views.generic import ListView
   ```

3. Import the `Joke` model from `models.py`, which is in the same directory:

   ```
   from .models import Joke
   ```

4. Create a `JokeListView` view that inherits from `ListView`:

   ```
   class JokeListView(ListView):
       model = Joke
   ```

   A minimal `ListView` is incredibly simple. It just requires the model to query.

The `jokes/views.py` file should now look like this:

Exercise Code 12.1: jokes/views.py

```
1.   from django.views.generic import ListView
2.
3.   from .models import Joke
4.
5.   class JokeListView(ListView):
6.       model = Joke
```

URLConf

You must now configure a path to the new view.

1. Open `jokes/urls.py` in your editor.

2. Import the `JokeListView` view from `views.py`, which is in the same directory:

```
from .views import JokeListView
```

3. Add a new path to `urlpatterns` to `JokeListView.as_view()` at `''`:

```
path('', JokeListView.as_view(), name='list'),
```

Remember that only URL paths that begin with `'/jokes/'` will be handed off to the URLConf of the `jokes` app, so `''` will actually be `'/jokes/'`. The second argument of the `path()` function must be a *view function* (as opposed to a class-based view), which is why you have to pass `JokeListView.as_view()`. The `as_view()` method of class-based views returns a view function.

The `jokes/urls.py` file should now look like this:

Exercise Code 12.2: jokes/urls.py

```
1.   from django.urls import path
2.
3.   from .views import JokeListView
4.
5.   app_name = 'jokes'
6.   urlpatterns = [
7.       path('', JokeListView.as_view(), name='list'),
8.   ]
```

Let's try it out.

1. If it's not still running, start up the server:

```
(.venv) …/projects/djangojokes.com> python manage.py runserver
```

2. Point your browser to `http://127.0.0.1:8000/jokes/`.
3. You should get an error that reads something like:

```
TemplateDoesNotExist at /jokes/
jokes/joke_list.html
```

Oops! We forgot to create a template for `JokeListView`. But, there is something even more interesting about this error: it tells us where it looked for that template: `jokes/joke_list.html`.

The Template

Take another look at the `JokeListView` code:

```
class JokeListView(ListView):
    model = Joke
```

Notice that you do not define a `template_name` attribute as you did with `HomePageView`. When no `template_name` is defined for a `ListView`, Django infers a `template_name` as:

```
app_name/model_list.html
```

where `app_name` is the name of the app (e.g., `jokes`) and model is the lowercase name of the model. So, for `JokeListView`, Django is looking for the template at:

```
jokes/joke_list.html
```

Create that template:

Create a `jokes` folder within the `templates` folder, and within that, create a `joke_list.html` file with the following content:

Exercise Code 12.3: templates/jokes/joke_list.html

```
1.   {% extends "_base.html" %}
2.
3.   {% block title %}Jokes{% endblock %}
4.   {% block main %}
5.     <h2>Jokes</h2>
6.     <ul class="list-group list-group-flush mb-3">
7.       {% for joke in joke_list %}
8.         <li class="list-group-item">{{ joke.question }}</li>
9.       {% endfor %}
10.    </ul>
11.  {% endblock %}
```

Things to notice:

1. The template extends `_base.html`:

```
{% extends "_base.html" %}
```

2. You use the `for` template tag, which we will cover in detail later (see page 131), to loop through the queryset, which is stored in an auto-created variable called `joke_list` (created by appending "_list" to the lowercase model name):

```
{% for joke in joke_list %}
<li class="list-group-item">{{ joke.question }}</li>
{% endfor %}
```

   The double curly braces hold a template variable (or expression) that should be evaluated. This for loop is the Django template equivalent of:

```
for joke in Joke.objects.all():
print(f'\n    <li class="list-group-item">{joke.question}</li>\n')
```

While you are working on templates, change the **NavLink** link in the header of the `_base.html` template to link to the jokes page using a `url` template tag:

```
<li class="nav-item">
  <a class="nav-link" href="{% url 'jokes:list' %}">Jokes</a>
</li>
```

Try It Out

In your browser, go to `http://127.0.0.1:8000/` and click the **Jokes** link in the header. It should look like this:

Right now, you only have one joke. You will add more soon, but first, you will create a `DetailView` for jokes.

Git Commit

Commit your code to Git.