Exercise 15: Deleting Jokes

15 to 25 minutes

In this exercise, you will create the view and form for deleting a joke:

1. Open `jokes/views.py` in your editor.
2. At the top of the page, import the `reverse_lazy()` function from `django.urls`:

   ```
   from django.urls import reverse_lazy
   ```

   `reverse_lazy()` works just like `reverse()`: it returns the URL based on the passed-in URL pattern name. But unlike `reverse()`, it waits to get the URL until it is needed. More on this in a moment.
3. Add `DeleteView` to the list of imported views from `django.views.generic`:

   ```
   from django.views.generic import (
       CreateView, DeleteView, DetailView, ListView, UpdateView
   )
   ```

4. Add the view:

   ```
   class JokeDeleteView(DeleteView):
       model = Joke
       success_url = reverse_lazy('jokes:list')
   ```

   Notice that we use `reverse_lazy()` here. The view is created before the URL configuration, so if you try to use `reverse()`, you will likely get an error about a circular import. The issue is that, with `reverse()`, the view needs the URLConf to have already been created, but the URLConf imports the view, so it cannot be created until the view exists.

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

Exercise Code 15.1: jokes/views.py

```python
from django.urls import reverse_lazy

from django.views.generic import (
    CreateView, DeleteView, DetailView, ListView, UpdateView
)

from .models import Joke

class JokeCreateView(CreateView):
    model = Joke
    fields = ['question', 'answer']


class JokeDeleteView(DeleteView):
    model = Joke
    success_url = reverse_lazy('jokes:list')


class JokeDetailView(DetailView):
    model = Joke


class JokeListView(ListView):
    model = Joke


class JokeUpdateView(UpdateView):
    model = Joke
    fields = ['question', 'answer']
```

URLConf

You must now configure the path to the new view. As you did with `UpdateView`, you will use the primary key to identify the joke to be deleted.

1. Open `jokes/urls.py` in your editor.
2. Add `JokeDeleteView` to the imported views:

```python
from .views import (
    JokeCreateView, JokeDeleteView, JokeDetailView, JokeListView, JokeUpdateView
)
```

3. Add a path for `JokeDeleteView` constructed as "`joke/<int:pk>/delete/`" and with the name "delete". It will resolve to something like "`jokes/joke/2/delete/`":

   ```
   path('joke/<int:pk>/delete/', JokeDeleteView.as_view(), name='delete'),
   ```

   "delete" is an arbitrary string. You could use "remove", "destroy" or anything else you like.

Exercise Code 15.2: jokes/urls.py

```
from django.urls import path

from .views import (
   JokeCreateView, JokeDeleteView, JokeDetailView, JokeListView, JokeUpdateView
)

app_name = 'jokes'
urlpatterns = [
    path('joke/<int:pk>/update/', JokeUpdateView.as_view(), name='update'),
    path('joke/<int:pk>/delete/', JokeDeleteView.as_view(), name='delete'),
    path('joke/create/', JokeCreateView.as_view(), name='create'),
    path('joke/<int:pk>/', JokeDetailView.as_view(), name='detail'),
    path('', JokeListView.as_view(), name='list'),
]
```

The Template

The default `template_name` for a `DeleteView` is inferred as follows:

```
app_name/model_confirm_delete.html
```

In this case, that's **`jokes/joke`**`_confirm_delete.html`.

Within the `templates/jokes` folder, add a `joke_confirm_delete.html` file with the following content:[19]

19. **Don't want to type?** Copy from `starter-code/app-with-model/joke_confirm_delete.html`.

Exercise Code 15.3: templates/jokes/joke_confirm_delete.html

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

{% block title %}Delete Joke{% endblock %}
{% block main %}
  <div class="card border-primary m-auto mb-3 text-center"
    style="max-width: 30rem">
    <form method="post">
      {% csrf_token %}
      <p><strong>Are you sure you want to delete this joke?</strong></p>
      <p>{{ joke.question }}</p>
      <button class="btn btn-success form-control">Confirm</button>
    </form>
  </div>
{% endblock %}
```

Things to notice:

1. The template must include a `form` element, and the method should be "post".
2. Within the `form` element, you must include:
   A. The `{% csrf_token %}` template tag – Again, this is a security measure.
   B. A submit button.

Open `templates/jokes/joke_list.html` and add a delete button link:

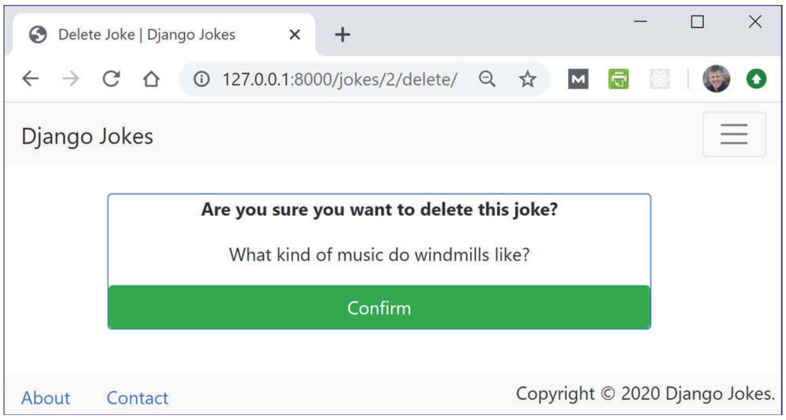
Exercise Code 15.4: templates/jokes/joke_list.html

```
-------Lines 1 through 11 Omitted-------
        <a href="{{ joke.get_absolute_url }}">{{ joke.question }}</a>
        <a href="{% url 'jokes:update' joke.pk %}"
          class="btn btn-info btn-sm float-right mr-2">Update</a>
        <a href="{% url 'jokes:delete' joke.pk %}"
          class="btn btn-danger btn-sm float-right mr-2">Delete</a>
-------Lines 17 through 20 Omitted-------
```

Note that you must pass the primary key with to the `delete` path so that the resulting page knows which joke to delete.

Try It Out

1. Visit `http://127.0.0.1:8000/jokes/`.
2. Click one of the **Delete** buttons. You should see a page like this:



Are you sure you want to delete this joke?

What kind of music do windmills like?

Copyright © 2020 Django Jokes.

3. Click **Confirm**. It should redirect to the **Jokes** page, and the joke you just deleted should be gone.

Git Commit

Commit your code to Git.

Conclusion

In this lesson, you have learned how models work and how to create a basic Django *CRUDL* (**C**reate, **R**ead, **U**pdate, **D**elete, **L**ist) app. You have also learned to use Git and GitHub for version control.

The completed jokes project for this lesson is available in `solutions/app-with-model/django jokes.com`.