Exercise 14: Creating and Updating Jokes

25 to 40 minutes

In this exercise, you will create the views and forms for creating and updating jokes. We will explain just the bare minimum about forms to get the pages working. Later, we will dig much deeper into Django forms (see page 219).

1. Open `jokes/views.py` in your editor.
2. Add `CreateView` and `UpdateView` to the list of imported views from `django.views.generic`:

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

3. Add two new views:

   A. `JokeCreateView`:

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

   B. `JokeUpdateView`:

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

   Notice that each of these views takes the model and the fields you want to include in the form.

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

Exercise Code 14.1: jokes/views.py

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

from .models import Joke

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


class JokeDetailView(DetailView):
    model = Joke


class JokeListView(ListView):
    model = Joke


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

URLConf

You must now configure paths to the new views. The `CreateView` view is used to create a *new* object (a joke), so its path will always be the same. But the path for the `UpdateView` view will be unique to each joke. For now, we will continue to use the primary key to identify the joke.

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

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

   Notice that we have wrapped the imported views in parentheses. This is so that we can break the list across lines to limit the length of each line to no longer than 79 characters as per PEP8.[16] We have also listed the imported views in alphabetical order, which makes things easier to find.

16. https://www.python.org/dev/peps/pep-0008/#maximum-line-length

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

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

4. Add a path for `JokeCreateView` constructed as "`joke/create/`" and with the name "create":

   ```
   path('joke/create/', JokeCreateView.as_view(), name='create'),
   ```

Note that "update" and "create" are arbitrary strings in the paths. You could use "edit" and "add" or anything else you like.

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

Exercise Code 14.2: jokes/urls.py

```
from django.urls import path

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

app_name = 'jokes'
urlpatterns = [
    path('joke/<int:pk>/update/', JokeUpdateView.as_view(), name='update'),
    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 `CreateView` and `UpdateView` views use the same default `template_name`, which they infer as follows:

```
app_name/model_form.html
```

In this case, that's `jokes/joke_form.html`.

Within the `templates/jokes` folder, add a `joke_form.html` file with the following content:[17]

Exercise Code 14.3: templates/jokes/joke_form.html

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

{% block title %}Add/Update Joke{% endblock %}
{% block main %}
  <div class="card border-primary m-auto mb-3 p-3"
    style="max-width: 30rem">
    <form method="post">
      {% csrf_token %}
      {{ form }}
      <button class="btn btn-success float-right">Submit</button>
    </form>
  </div>
{% endblock %}
```

Things to notice:

1. The template must include a `form` element, and the method must be "post".
2. Within the `form` element, you must include:
   A. The `{% csrf_token %}` template tag – This is a security measure that protects against cross site request forgery.[18]
   B. The `form` variable: `{{ form }}` – This will output the form fields and their labels.
   C. A submit button.

Open `templates/jokes/joke_list.html` and add links to the new views. Use Bootstrap to make the links look like buttons:

17. **Don't want to type?** Copy from `starter-code/app-with-model/joke_form.html`.
18. https://docs.djangoproject.com/en/3.1/ref/csrf/

Exercise Code 14.4: templates/jokes/joke_list.html

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

{% block title %}Jokes{% endblock %}
{% block main %}
  <a class="btn btn-success btn-sm float-right" href="{% url 'jokes:create' %}">
    + New Joke
  </a>
  <h2>Jokes</h2>
  <ul class="list-group list-group-flush mb-3">
    {% for joke in joke_list %}
      <li class="list-group-item">
        <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>
      </li>
    {% endfor %}
  </ul>
{% endblock %}
```

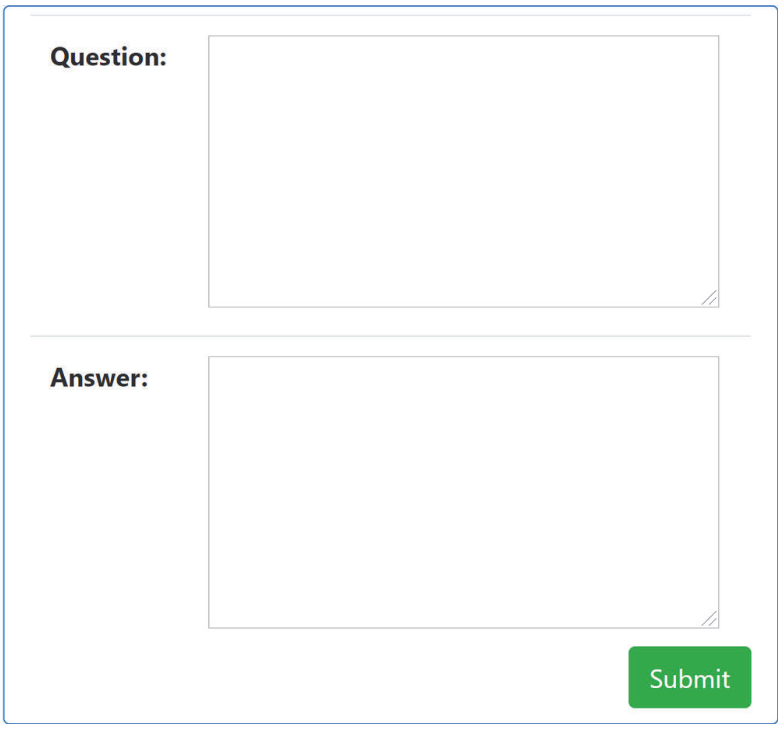
Note that you must pass the primary key to the `jokes:update` path so that the resulting page knows which joke to update.

Visit `http://127.0.0.1:8000/jokes/` and click the **+ New Joke** button. You should see the form, but it won't be very pretty. You can make it look better by using `form.as_table` and nesting the `form` in a `table`:

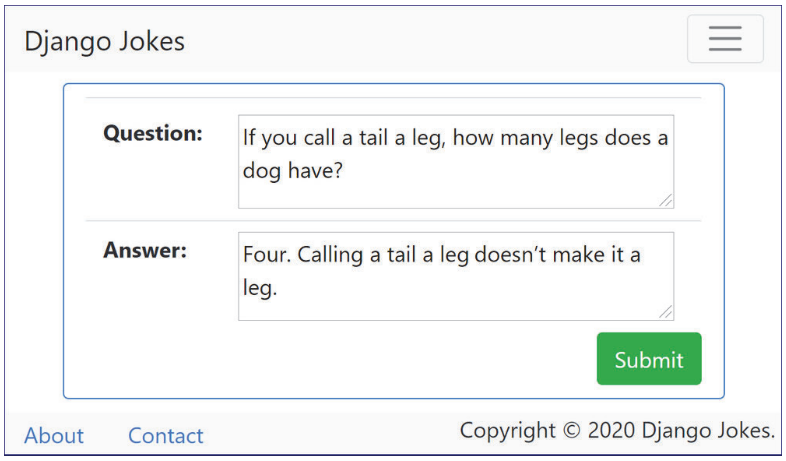
Exercise Code 14.5: templates/jokes/joke_form.html

```
-------Lines 1 through 6 Omitted-------
7.        <form method="post">
8.          {% csrf_token %}
9.          <table class="table">
10.           {{ form.as_table }}
11.         </table>
12.         <button class="btn btn-success float-right">Submit</button>
13.       </form>
-------Lines 14 through 15 Omitted-------
```

Refresh the page with the create-joke form. It should now look like this:



That's better. We'll make additional improvements later. Now, go ahead and submit a joke:

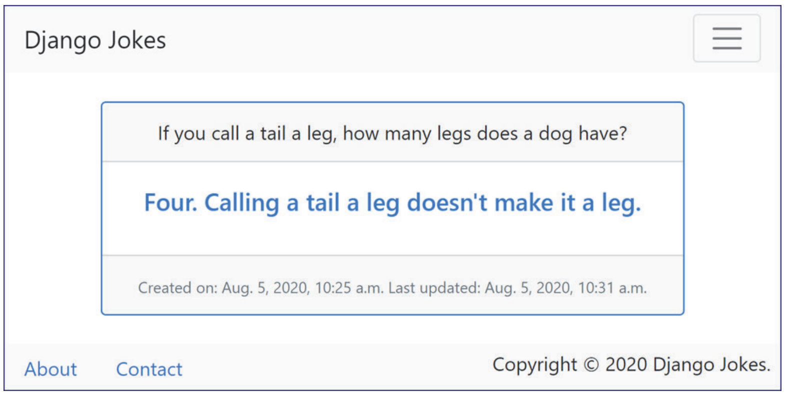


If you call a tail a leg, how many legs does a dog have?

Four. Calling a tail a leg doesn't make it a leg.

Copyright © 2020 Django Jokes.

You should be redirected to the detail page for the new joke:



Django Jokes

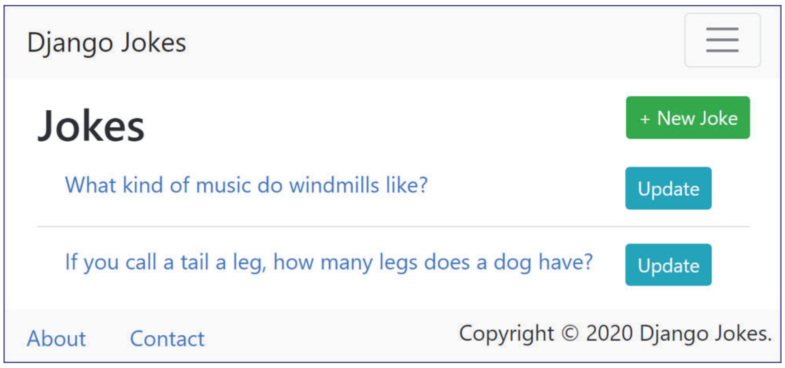
If you call a tail a leg, how many legs does a dog have?

Four. Calling a tail a leg doesn't make it a leg.

Created on: Aug. 5, 2020, 10:25 a.m. Last updated: Aug. 5, 2020, 10:31 a.m.

About Contact Copyright © 2020 Django Jokes.

Go to the **Jokes** page. It should show your new joke:



Django Jokes

Jokes

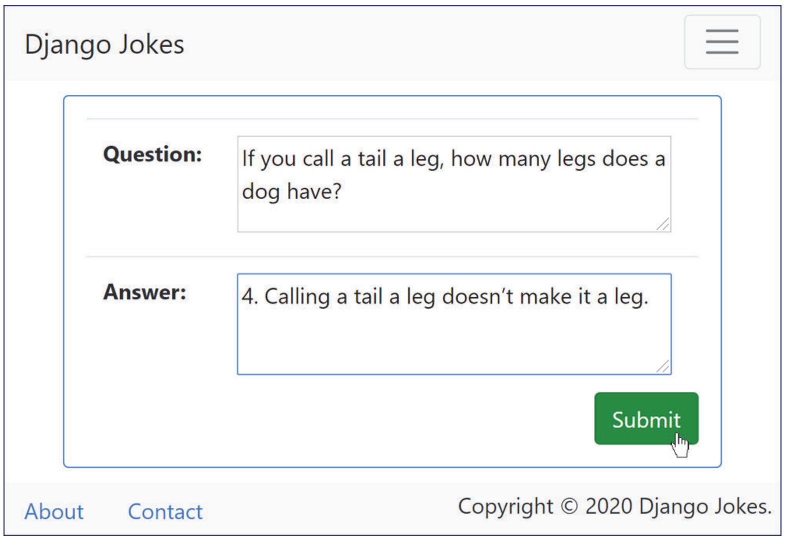
+ New Joke

What kind of music do windmills like? Update

If you call a tail a leg, how many legs does a dog have? Update

About Contact Copyright © 2020 Django Jokes.

Click the **Update** button and update the joke:



Django Jokes

Question: If you call a tail a leg, how many legs does a dog have?

Answer: 4. Calling a tail a leg doesn't make it a leg.

Submit

About Contact Copyright © 2020 Django Jokes.

You can now add and update jokes.

Git Commit

Commit your code to Git.

2.6. Deleting Objects

Django's default delete process works like this:

1. The user clicks a **Delete** link or button, resulting in a *get* request to a **Delete Confirmation** page, which contains a simple form with a single submit button.
2. The user submits the form, resulting in a *post* request that deletes the record from the database.
3. The server redirects to a "success url" specified in the view as `success_url`.