Exercise 28: Changing Jokes to Use Slugs

30 to 45 minutes

In this exercise, you will change the `jokes` app to use slugs instead of ids in URLs.

❖ E28.1. The Model

1. Open `jokes/models.py` in your editor:

    A. Import `unique_slug` from `common.utils.text`:

    ```
    from common.utils.text import unique_slug
    ```

    B. Add a `slug` field to the model:

    ```
    slug = models.SlugField(
        max_length=50, unique=True, null=True, editable=False
    )
    ```

    `editable` is set to `False` because you don't want this field to be editable in Django admin (or anywhere else).[49] Also, remember that you are just temporarily allowing `null` values.

2. You now need to override the model's `save()` method. Remember when you did the following at the Django shell to save a joke object:

```
>>> from jokes.models import Joke
>>> q = 'Why did the chicken cross the road?'
>>> a = 'To get to the other side.'
>>> joke = Joke(question=q, answer=a)
>>> joke.save()
```

49. You may actually want slugs to be editable in Django admin. It depends whether you want admins to be able to modify the URL of a page. If you do, remove `editable=False`.

Before the object's `save()` method is called, you need to set the value of `slug`. To do that, you need to override the `save()` method in the `Joke` model, like this:

```
def save(self, *args, **kwargs):
    if not self.slug:
        value = str(self)
        self.slug = unique_slug(value, type(self))

    super().save(*args, **kwargs)
```

Let's look at this line by line:

A. Only create the slug if the record doesn't already have one:

   ```
   if not self.slug:
   ```

   This will be the case for *all new records* and for jokes that were added before you changed the model.

B. Set `value` to the value returned by the `__str__()` method:

   ```
       value = str(self)
   ```

   You could set this explicitly to `self.question`, but doing it this way makes the method reusable *as is* in other models.

C. Assign the unique slug to the object:

   ```
       self.slug = unique_slug(value, type(self))
   ```

   `type(self)` gets the class of this object. You could explicitly use `Joke` here, but doing it this way makes the method reusable *as is* in other models.

D. Call `super().save()` to do whatever the `save()` method of `models.Model` does[50] to save the object:

   ```
   super().save(*args, **kwargs)
   ```

   If you don't do this, the object won't get saved.

50. To see what the `save()` method of `models.Model` does, open `db/models/base.py` from the `django` library and search for "def save". It does quite a lot!

3. Finally, you need to fix the `get_absolute_url()` function to use `slug` instead of `id`:

```
def get_absolute_url(self):
    return reverse('joke', args=[self.slug])
```

The complete code looks like this:

Exercise Code 28.1: jokes/models.py

```
from django.db import models
from django.urls import reverse

from common.utils.text import unique_slug

class Joke(models.Model):
    question = models.TextField(max_length=200)
    answer = models.TextField(max_length=100, blank=True)
    slug = models.SlugField(
        max_length=50, unique=True, null=True, editable=False
    )
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)

    def get_absolute_url(self):
        return reverse('jokes:detail', args=[self.slug])

    def save(self, *args, **kwargs):
        if not self.slug:
            value = str(self)
            self.slug = unique_slug(value, type(self))

        super().save(*args, **kwargs)

    def __str__(self):
        return self.question
```

❖ E28.2. Migrating

As you have changed the model, you need to make and run migrations:

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

❖ E28.3. Django admin

Because you set `editable` to `False` for the `slug` field, `slug` won't show up in Django admin. To see the `slug` value when editing a joke, add `slug` to the returned read-only fields in `jokes/admin.py`:

```
if obj: # editing an existing object
    return ('slug', 'created', 'updated')
```

❖ E28.4. Reconfiguring the URLs

You need to make the URL patterns for joke pages use the slug. The current URLConf looks like this:

Exercise Code 28.2: 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'),
]
```

Replace every instance of `<int:pk>` with `<slug>`:

Exercise Code 28.3: jokes/urls.py

```
    -------Lines 1 through 8 Omitted-------
    urlpatterns = [
        path('joke/<slug>/update/', JokeUpdateView.as_view(), name='update'),
        path('joke/<slug>/delete/', JokeDeleteView.as_view(), name='delete'),
        path('joke/create/', JokeCreateView.as_view(), name='create'),
        path('joke/<slug>/', JokeDetailView.as_view(), name='detail'),
        path('', JokeListView.as_view(), name='list'),
    ]
```

Notice that you do not need to preface `slug` with `str:` as you do with `<int:pk>`. All values coming in over the URL are already strings.

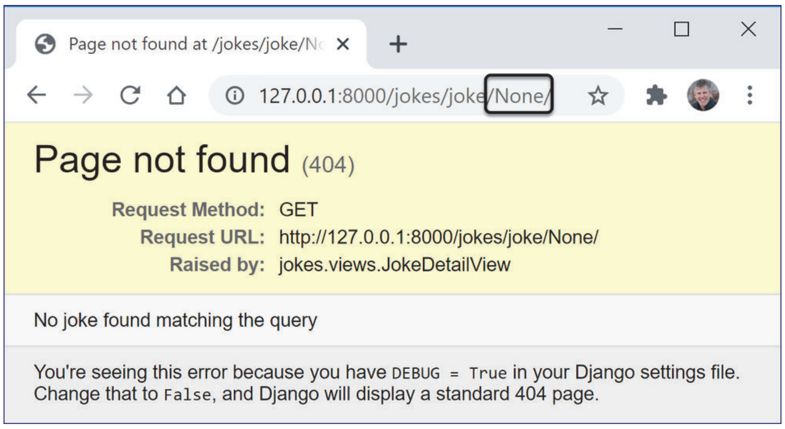
The view will have access to the slug value via `self.kwargs.get('slug')` and will use it to get the joke object to use.

❖ E28.5. Viewing Jokes

1. Start up the server:

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

2. Visit the site, click the **Jokes** link in the header, and then on one of the jokes. You should get a 404 error:

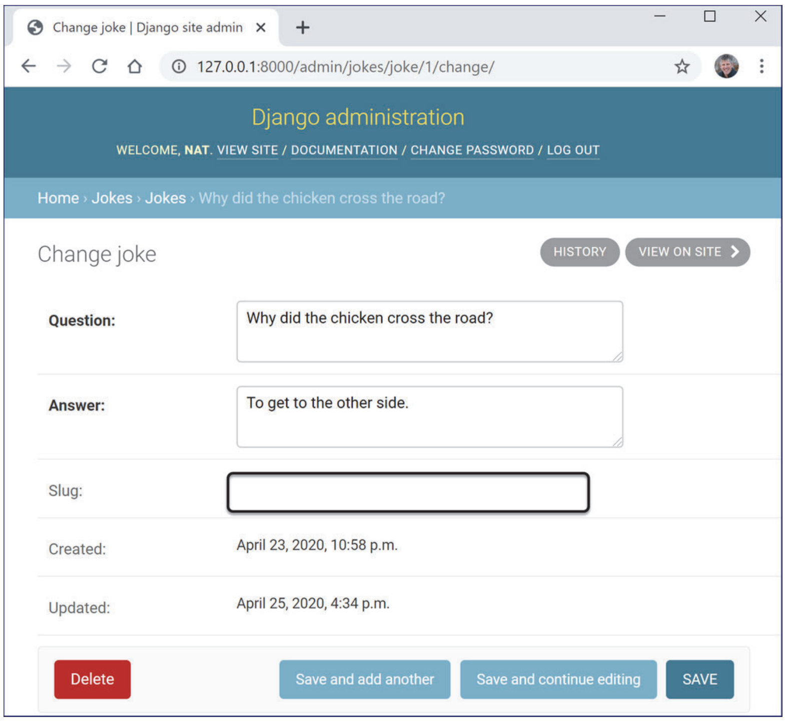


```
Page not found (404)
      Request Method:  GET
         Request URL:  http://127.0.0.1:8000/jokes/joke/None/
           Raised by:  jokes.views.JokeDetailView

No joke found matching the query

You're seeing this error because you have DEBUG = True in your Django settings file.
Change that to False, and Django will display a standard 404 page.
```

Page not found at /jokes/joke/No
127.0.0.1:8000/jokes/joke/None/

This is because existing jokes don't have slugs yet. You need to add them.

3. In `Django admin (http://127.0.0.1:8000/admin/)`, open up a joke:



Change joke | Django site admin
127.0.0.1:8000/admin/jokes/joke/1/change/
Django administration
WELCOME, NAT. VIEW SITE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT
Home › Jokes › Jokes › Why did the chicken cross the road?
Change joke
HISTORY
VIEW ON SITE
Delete
Save and add another
Save and continue editing
SAVE

Question:	Why did the chicken cross the road?
Answer:	To get to the other side.
Slug:	
Created:	April 23, 2020, 10:58 p.m.
Updated:	April 25, 2020, 4:34 p.m.

Notice the slug field is empty.

4. Click the **Save and continue editing** button. The slug should now show up as something like:

```
Slug:              why-did-the-chicken-cross-the-road
```

❖ E28.6. Saving All Existing Jokes

You could open and save each joke individually in Django admin, which would be easy in this case, because you don't have many jokes yet. However, if you did have a lot of jokes, you would want to do this at the shell, so let's learn to do it that way:

1. Open the shell:

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

2. Import the Joke model:

```
>>> from jokes.models import Joke
```

3. Loop through all the jokes, saving each one and then exit the shell:

```
>>> for joke in Joke.objects.all():
...     joke.save()
...
>>> exit()
```

4. Restart the server if necessary:

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

5. Return to the site, click the **Jokes** link in the header, and then on one of the jokes. The page should now come up and the URL should be something like:

```
http://127.0.0.1:8000/jokes/joke/why-did-the-chicken-cross-the-road/
```

❖ E28.7. Setting `null` to `False` for the `SlugField`

Now that you have updated all the `slug` values, change the value of `null` to `False` for `slug`:

```
slug = models.SlugField(
    max_length=50, unique=True, null=False, editable=False
)
```

As you have changed the model, you will need to make and run migrations:

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

When you run this you will get a warning and a message with various options. The warning will say something like "You are trying to change the nullable field 'slug' on joke to non-nullable without a default; we can't do that." Django doesn't appear to know that you've already dealt with all the `null` values. Since you have, you can safely precede, but you will have to choose one of the options. Enter `1` to provide a one-off default and then enter `'foo'` (in quotes) or any random string for the default:

```
Select an option: 1
Please enter the default value now, as valid Python
The datetime and django.utils.timezone modules are available, so you can do e.g. time ↵↵
zone.now
Type 'exit' to exit this prompt
>>> 'foo'
Migrations for 'jokes':
  jokes\migrations\0004_auto_20200425_2147.py
    - Alter field slug on joke
```

You have made the migrations script. Now, you can run migrations:

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

❖ E28.8. Update Joke List Template

The joke list template (`templates/jokes/joke_list.html`), currently still uses `joke.pk` for the links to **Update** and **Delete**:

```
<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>
```

Change those both to `joke.slug`.

❖ E28.9. Add a New Joke

1. Go to the Jokes listing and click the **+ New Joke** button.
2. Add a new joke and notice that the URL that you're redirected to for the new joke uses the joke's slug.
3. Add this joke (be sure to include spaces before and after the plus sign):
   - Question: What is 1 + 1?
   - Answer: 11

   Notice that the slug is `what-is-1-1`.
4. Now, add this joke:

- Question: What is 1 - 1?
- Answer: H

Notice that the slug ends with a random string of characters (e.g., `what-is-1-1-jemfemkhxx`).

While neither of these last two jokes is funny, they do demonstrate how the `unique_slug()` function prevents duplicate slugs by adding a random string.

Play around with the site some by adding, updating, and deleting jokes.

Git Commit

Commit your code to Git.

Conclusion

In this lesson, you have learned to create slugs and to override the `save()` method in your models, so that you can autogenerate slugs when a new record is created. You have also learned how to make changes to an existing model.