Chapter 13: Newspaper App

حان الوقت لإنشاء تطبيق الصحف الخاص بنا. سيكون لدينا صفحة مقالات حيث يمكن للصحفيين نشر المقالات ، وإعداد الأذونات بحيث لا يتمكن سوى مؤلف المقالة من تحريرها أو حذفها ، وأخيرًا إضافة إمكانية للمستخدمين الآخرين لكتابة تعليقات على كل مقالة والتي ستقدم مفهوم الأجنبية مفاتيح. تطبيق Articles لبدء إنشاء تطبيق مقالات وتحديد نماذج قاعدة البيانات الخاصة بنا. لا توجد قواعد صارمة وسريعة حول تسمية تطبيقاتك باستثناء أنه لا يمكنك استخدام اسم تطبيق مدمج. إذا نظرت إلى قسم INSTALLED_APPS من config / settings.py ، يمكنك معرفة أسماء التطبيقات المحظورة: admin ، والمصادقة ، وأنواع المحتوى ، والجلسات ، والرسائل ، والملفات الثابتة. تتمثل القاعدة العامة في استخدام صيغة الجمع لاسم التطبيق - المنشورات ، والمدفوعات ، والمستخدمون ، وما إلى ذلك - ما لم يكن القيام بذلك خطأ كما هو الحال في الحالة العامة للمدونة حيث يكون المفرد أكثر منطقية. ابدأ بإنشاء تطبيق المقالات الجديد الخاص بنا

Command Line

(news) $ python manage.py startapp articles

ثم أضفه إلى INSTALLED_APPS وقم بتحديث المنطقة الزمنية لأننا سنقوم بوضع طابع زمني لمقالاتنا. يمكنك العثور على منطقتك الزمنية في قائمة ويكيبيديا هذه 159. على سبيل المثال ، أعيش في بوسطن ، ماساتشوستس الواقعة في المنطقة الزمنية الشرقية للولايات المتحدة. لذلك دخولي هو America / New_York.

Code

```
# config/settings.py
INSTALLED_APPS = [
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
# 3rd Party
'crispy_forms',
# Local
'accounts',
```

```
'pages',
'articles', # new
]
TIME_ZONE = 'America/New_York' # new
```

بعد ذلك نحدد نموذج قاعدة البيانات الخاص بنا والذي يحتوي على أربعة حقول: العنوان والجسم والتاريخ والمؤلف.
لاحظ أننا نسمح لـ Django بتعيين الوقت والتاريخ تلقائيًا بناءً على إعداد TIME_ZONE الخاص بنا. بالنسبة لحقل المؤلف ، نريد أن نشير إلى نموذج المستخدم المخصص 'accounts.CustomUser' الذي قمنا بتعيينه في ملف config / settings.py كـ AUTH_USER_MODEL. يمكننا القيام بذلك عبر get_user_model161.
ونطبق أيضًا أفضل الممارسات لتحديد get_absolute_url من البداية وطريقة __str__ لعرض النموذج في واجهة الإدارة الخاصة بنا.

Code

```
# articles/models.py
from django.conf import settings
from django.contrib.auth import get_user_model
from django.db import models
from django.urls import reverse
class Article(models.Model):
title = models.CharField(max_length=255)
body = models.TextField()
date = models.DateTimeField(auto_now_add=True)
author = models.ForeignKey(
get_user_model(),
on_delete=models.CASCADE,
)
def __str__(self):
return self.title
def get_absolute_url(self):
return reverse('article_detail', args=[str(self.id)])
```

نظرًا لأن لدينا تطبيقًا ونموذجًا جديدين تمامًا ، فقد حان الوقت لإنشاء ملف ترحيل جديد ثم تطبيقه على قاعدة البيانات.

Command Line

(news) $ python manage.py makemigrations articles

(news) $ python manage.py migrate

في هذه المرحلة ، أود الانتقال إلى المسؤول للتلاعب بالنموذج قبل إنشاء عناوين

urls/views/templates

اللازمة لعرض البيانات فعليًا على موقع الويب. لكن نحتاج أولاً إلى تحديث articles / admin.py حتى يتم عرض تطبيقنا الجديد

Code

```
# articles/admin.py
from django.contrib import admin
from .models import Article
admin.site.register(Article)
```

Now we start the server.

Command Line

(news) $ python manage.py runserver

Navigate to http://127.0.0.1:8000/admin/ and log in.

إذا قمت بالنقر فوق "+ إضافة" بجوار "مقالات" في الجزء العلوي من الصفحة ، فيمكننا إدخال بعض عينات البيانات.
من المحتمل أن يكون لديك ثلاثة مستخدمين متاحين في هذه المرحلة:استخدم حساب admin الخاص بك كمؤلف لجميع المقالات الثلاثة.
لقد أضفت ثلاث مقالات جديدة كما ترى في صفحة المقالات المحدثة.

إذا قمت بالنقر فوق مقال فردي ، فسترى أنه يتم عرض العنوان والجسم والمؤلف ولكن ليس التاريخ. هذا لأن التاريخ تمت إضافته تلقائيًا بواسطة Django وبالتالي لا يمكن تغييره في المشرف. يمكننا أن نجعل التاريخ قابلاً للتعديل - في التطبيقات الأكثر تعقيدًا ، من الشائع أن يكون لديك حقلاً مُنشأ ومُحدَّث - ولكن لتبسيط الأمور ، سنحدد التاريخ عند الإنشاء بواسطة Django بالنسبة لنا في الوقت الحالي. على الرغم من عدم عرض التاريخ هنا ، فلا يزال بإمكاننا الوصول إليه في قوالبنا حتى يمكن عرضه على صفحات الويب. عناوين URL وطرق العرض الخطوة التالية هي

تكوين عناوين URL وطرق العرض الخاصة بنا. دعنا تظهر مقالاتنا في المقالات /. أضف نمط URL للمقالات في ملف config / urls.py الخاص بنا

Code

```
# config/urls.py
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
path('admin/', admin.site.urls),
path('accounts/', include('accounts.urls')),
path('accounts/', include('django.contrib.auth.urls')),
path('articles/', include('articles.urls')), # new
path('', include('pages.urls')),
]
```

Next we create an articles/urls.py file.

Command Line

```
(news) $ touch articles/urls.py
```

ثم املأها بمساراتنا. لنبدأ بالصفحة لسرد جميع المقالات في المقالات / التي ستستخدم طريقة العرضArticleListView.

Code

```
# articles/urls.py
from django.urls import path
from .views import ArticleListView
urlpatterns = [
path('', ArticleListView.as_view(), name='article_list'),
]
```

الآن قم بإنشاء view باستخدام ListView العام المدمج من Django.

Code

```
# articles/views.py
from django.views.generic import ListView
from .models import Article
class ArticleListView(ListView):
model = Article
template_name = 'article_list.html'
```

المجالان الوحيدان اللذان نحتاج إلى تحديدهما هما مقالة النموذج واسم النموذج الخاص بنا والذي سيكون article_list.html. الخطوة الأخيرة هي إنشاء النموذج الخاص بنا. يمكننا إنشاء ملف فارغ من سطر الأوامر.

Command Line

```
(news) $ touch templates/article_list.html
```

يحتوي Bootstrap على مكون مدمج يسمى Cards يمكننا تخصيصه لمقالاتنا الفردية. تذكر أن ListView تقوم بإرجاع كائن يسمى object_list والذي يمكننا التكرار عليه باستخدام حلقة for. في كل مقال نعرض title و body و author و والتاريخ. يمكننا أيضًا توفير روابط لوظيفة "تعديل" و "حذف" لم نقم بإنشائها بعد

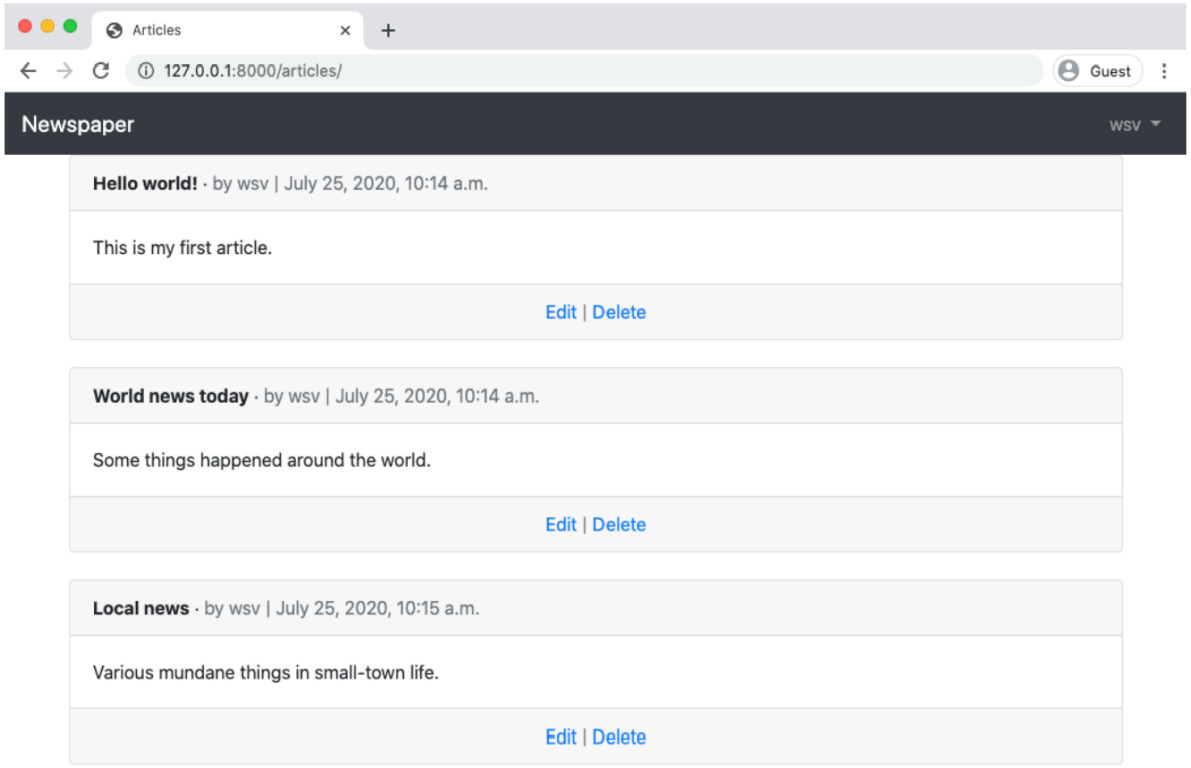
Code

```
<!-- templates/article_list.html -->
{% extends 'base.html' %}
{% block title %}Articles{% endblock title %}
{% block content %}
{% for article in object_list %}
<div class="card">
<div class="card-header">
<span class="font-weight-bold">{{ article.title }}</span> &middot;
<span class="text-muted">by {{ article.author }} |
{{ article.date }}</span>
</div>
<div class="card-body">
```

```
{{ article.body }}
</div>
<div class="card-footer text-center text-muted">
<a href="#">Edit</a> | <a href="#">Delete</a>
</div>
</div>
<br />
{% endfor %}
{% endblock content %}
```

قم بتدوير الخادم مرة أخرى باستخدام python manager.py runserver وتحقق من صفحتنا على

http://127.0.0.1:8000/articles/



Articles page

ليس سيئا إيه؟ إذا أردنا أن نكون خياليين ، فيمكننا إنشاء عامل تصفية قالب مخصص بحيث يظهر التاريخ الناتج بالثواني أو الدقائق أو الأيام. يمكن القيام بذلك مع بعض منطق if / else وخيارات تاريخ Django لكننا لن ننفذها هنا. تحرير /

حذف كيف نضيف خيارات التعديل والحذف؟ نحتاج إلى عناوين url وطرق عرض وقوالب جديدة. لنبدأ بعناوين url.
يمكننا الاستفادة من حقيقة أن Django يضيف تلقائيًا مفتاحًا أساسيًا إلى كل قاعدة بيانات. لذلك ستكون مقالتنا الأولى ذات المفتاح الأساسي 1 في

articles/1/edit/ and the delete route will be at articles/1/delete/

Code

```
# articles/urls.py
from django.urls import path
from .views import (
ArticleListView,
ArticleUpdateView, # new
ArticleDetailView, # new
ArticleDeleteView, # new
)
urlpatterns = [
path('<int:pk>/edit/',
ArticleUpdateView.as_view(), name='article_edit'), # new
path('<int:pk>/',
ArticleDetailView.as_view(), name='article_detail'), # new
path('<int:pk>/delete/',
ArticleDeleteView.as_view(), name='article_delete'), # new
path('', ArticleListView.as_view(), name='article_list'),
]
```

اكتب الآن وجهات نظرنا التي ستستخدم طرق عرض Django العامة المستندة إلى الفصل لـ DetailView و UpdateView و DeleteView. نحدد الحقول التي يمكن تحديثها - العنوان والجسم - ومكان إعادة توجيه المستخدم بعد حذف المقالة: article_list

Code

```
# articles/views.py
from django.views.generic import ListView, DetailView # new
from django.views.generic.edit import UpdateView, DeleteView # new
```

```
from django.urls import reverse_lazy # new
from .models import Article
class ArticleListView(ListView):
model = Article
template_name = 'article_list.html'
class ArticleDetailView(DetailView): # new
model = Article
template_name = 'article_detail.html'
class ArticleUpdateView(UpdateView): # new
model = Article
fields = ('title', 'body',)
template_name = 'article_edit.html'
class ArticleDeleteView(DeleteView): # new
model = Article
template_name = 'article_delete.html'
success_url = reverse_lazy('article_list')
```

أخيرًا ، نحتاج إلى إضافة قوالبنا الجديدة. أوقف الخادم باستخدام Control + c واكتب ما يلي.

Command Line

```
(news) $ touch templates/article_detail.html
(news) $ touch templates/article_edit.html
(news) $ touch templates/article_delete.html
```

سنبدأ بصفحة التفاصيل التي ستعرض العنوان والتاريخ والجسم والمؤلف مع روابط للتعديل والحذف. كما سيتم ربط العودة إلى جميع المقالات. تذكر أن علامة url الخاصة بلغة Django النموذجية تريد اسم URL ثم أي وسيطات تم تمريرها. اسم مسار التحرير الخاص بنا هو article_edit ونحتاج إلى تمرير مفتاحه الأساسي article.pk. اسم مسار الحذف هو article_delete ويحتاج أيضًا إلى article.pk مفتاح أساسي. صفحة المقالات الخاصة بنا هي ListView لذا فهي لا تحتاج إلى تمرير أية وسيطات إضافية.

Code

```
<!-- templates/article_detail.html -->
{% extends 'base.html' %}
```

```
{% block content %}
<div class="article-entry">
<h2>{{ object.title }}</h2>
<p>by {{ object.author }} | {{ object.date }}</p>
<p>{{ object.body }}</p>
</div>
<p><a href="{% url 'article_edit' article.pk %}">Edit</a> |
<a href="{% url 'article_delete' article.pk %}">Delete</a></p>
<p>Back to <a href="{% url 'article_list' %}">All Articles</a>.</p>
{% endblock content %}
```

لتحرير الصفحات وحذفها ، يمكننا استخدام نمط زر Bootstrap لجعل زر التحرير أزرق فاتح وزر الحذف أحمر.

Code

```
<!-- templates/article_edit.html -->
{% extends 'base.html' %}
{% block content %}
<h1>Edit</h1>
<form action="" method="post">{% csrf_token %}
{{ form.as_p }}
<button class="btn btn-info ml-2" type="submit">Update</button>
</form>
{% endblock content %}
```

Code

```
<!-- templates/article_delete.html -->
{% extends 'base.html' %}
{% block content %}
<h1>Delete</h1>
<form action="" method="post">{% csrf_token %}
<p>Are you sure you want to delete "{{ article.title }}"?</p>
```

<button class="btn btn-danger ml-2" type="submit">Confirm</button>

</form>

{% endblock content %}

كخطوة أخيرة ، في قسم تذييل البطاقة في article_list.html ، يمكننا استبدال "العنصر النائب a hrefs بدلاً من مسارات عنوان URL الفعلية ، باستخدام علامة قالب url166 ، واسم عنوان URL ، وكمعامل pk لكل منها .

Code

<!-- templates/article_list.html -->

...

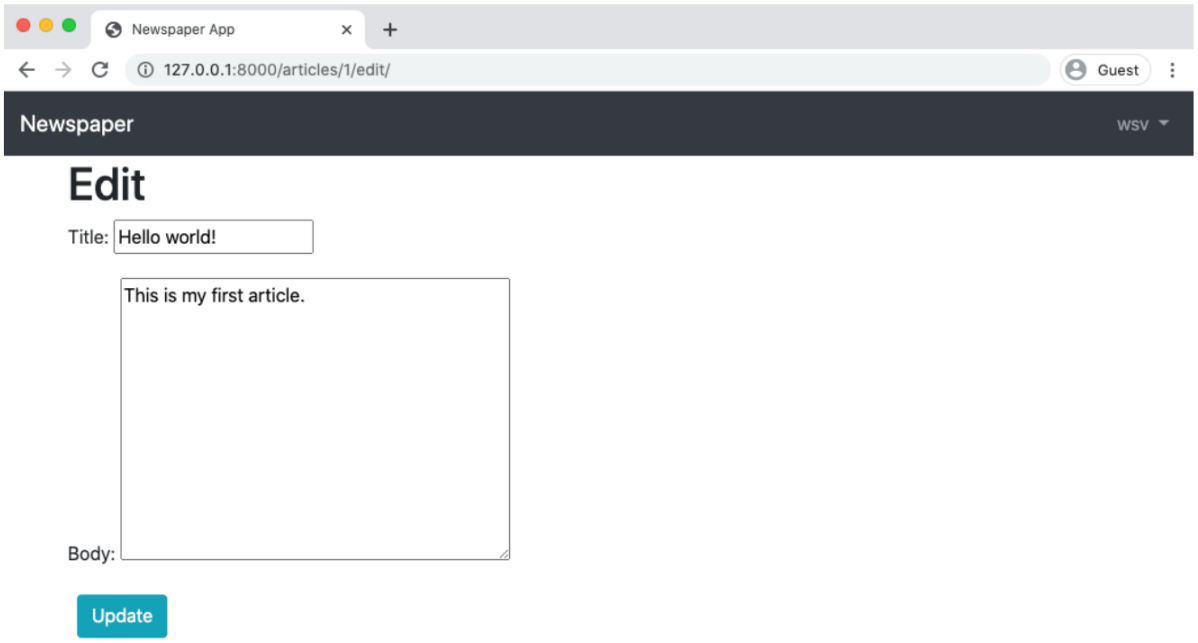
Edit |

Delete

…

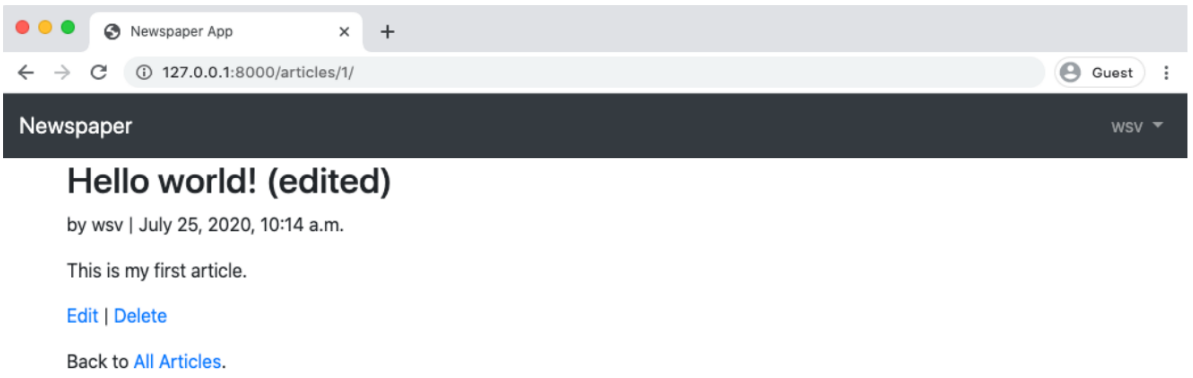
حسنًا ، نحن جاهزون لعرض عملنا. ابدأ تشغيل الخادم باستخدام python manager.py runserver وانتقل إلى صفحة المقالات على http://127.0.0.1:8000/articles/. انقر على رابط "تعديل" في المقالة الأولى وستتم إعادة توجيهك إلى

http://127.0.0.1:8000/articles/1/edit/



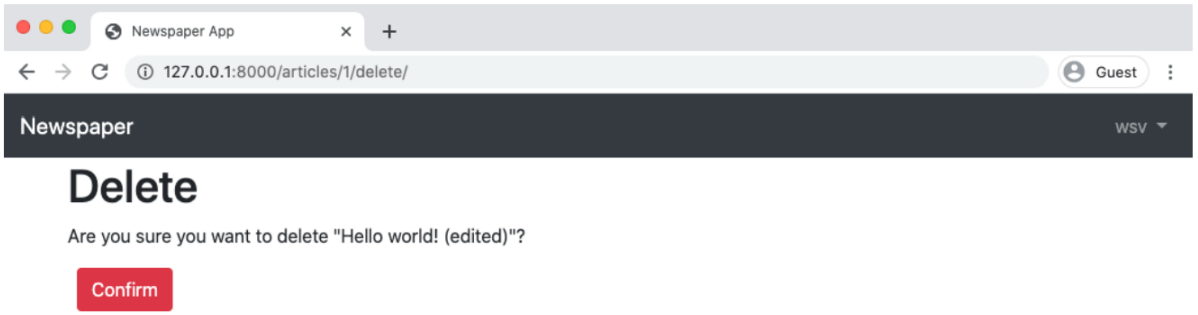
Edit page

إذا قمت بتحديث حقل "العنوان" عن طريق إضافة "(محرر)" في النهاية والنقر فوق تحديث ، فسيتم إعادة توجيهك إلى صفحة التفاصيل التي تعرض التغيير الجديد.



Detail page

إذا نقرت على رابط "حذف" ، فسيتم إعادة توجيهك إلى صفحة الحذف



Delete page

اضغط على الزر الأحمر المخيف لـ "حذف" وستتم إعادة توجيهك إلى صفحة المقالات التي تحتوي الآن على إدخالين فقط

Create Page

الخطوة الأخيرة هي إنشاء صفحة للمقالات الجديدة والتي يمكننا القيام بها باستخدام برنامج CreateView لـ Django. تتمثل خطواتنا الثلاث في إنشاء عرض وعنوان URL ونموذج. يجب أن يشعر هذا التدفق مألوفًا جدًا الآن. في ملف العروض الخاص بنا ، أضف CreateView إلى الواردات في الجزء العلوي وقم بإنشاء فئة جديدة في أسفل الملف articleCreateView تحدد النموذج والقالب والحقول المتاحة

Code

```
# articles/views.py
...
from django.views.generic.edit import (
UpdateView, DeleteView, CreateView # new
)
...
class ArticleCreateView(CreateView): # new
model = Article
template_name = 'article_new.html'
fields = ('title', 'body', 'author',)
```

لاحظ أن حقولنا لها مؤلف لأننا نريد ربط مقال جديد بمؤلف ، ولكن بمجرد إنشاء مقال ، لا نريد أن يكون المستخدم قادرًا على تغيير المؤلف وهذا هو السبب في أن ArticleUpdateView لديه الحقول فقط ['title', 'body',]. قم بتحديث ملف عناوين url الخاص بنا بالمسار الجديد view.

Code

```
# articles/urls.py
from django.urls import path
from .views import (
ArticleListView,
ArticleUpdateView,
ArticleDetailView,
ArticleDeleteView,
ArticleCreateView, # new
)
urlpatterns = [
path('<int:pk>/edit/',
ArticleUpdateView.as_view(), name='article_edit'),
path('<int:pk>/',
ArticleDetailView.as_view(), name='article_detail'),
path('<int:pk>/delete/',
ArticleDeleteView.as_view(), name='article_delete'),
path('new/', ArticleCreateView.as_view(), name='article_new'), # new
path('', ArticleListView.as_view(), name='article_list'),
]
```

ثم قم بإنهاء الخادم Control + c لإنشاء قالب جديد باسم

article_new.html

Command Line

```
(news) $ touch templates/article_new.html
```

And update it with the following HTML code.

Code

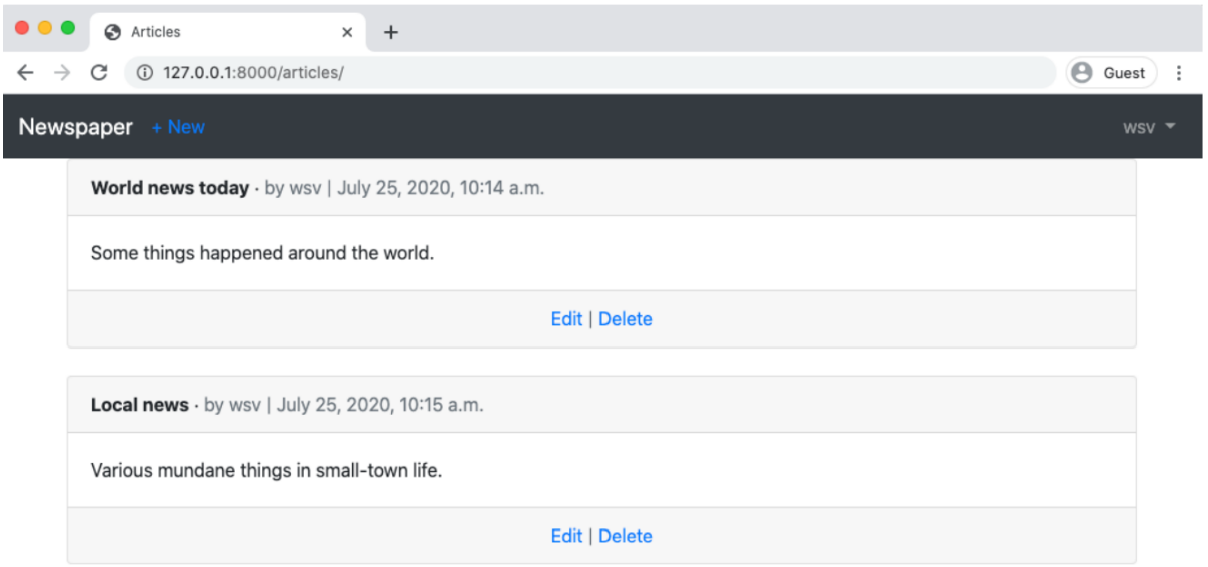
```
<!-- templates/article_new.html -->
{% extends 'base.html' %}
{% block content %}
<h1>New article</h1>
<form action="" method="post">{% csrf_token %}
{{ form.as_p }}
<button class="btn btn-success ml-2" type="submit">Save</button>
</form>
{% endblock content %}
```

أخيرًا ، يجب أن نضيف رابطًا لإنشاء مقالات جديدة في شريط التنقل الخاص بنا بحيث يمكن الوصول إليه في كل مكان على الموقع للمستخدمين الذين قاموا بتسجيل الدخول.

Code

```
<!-- templates/base.html -->
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="{% url 'home' %}">Newspaper</a>
{% if user.is_authenticated %}
<ul class="navbar-nav mr-auto">
<li class="nav-item">
<a href="{% url 'article_new' %}">+ New</a>
</li>
</ul>
{% endif %}
...
```

إذا كنت بحاجة إلى مساعدة للتأكد من دقة ملف HTML الخاص بك الآن ، فيرجى الرجوع إلى كود المصدر الرسمي 167. قم بتحديث صفحة المقالات وسيظهر التغيير في شريط التنقل العلوي:



Navbar new link

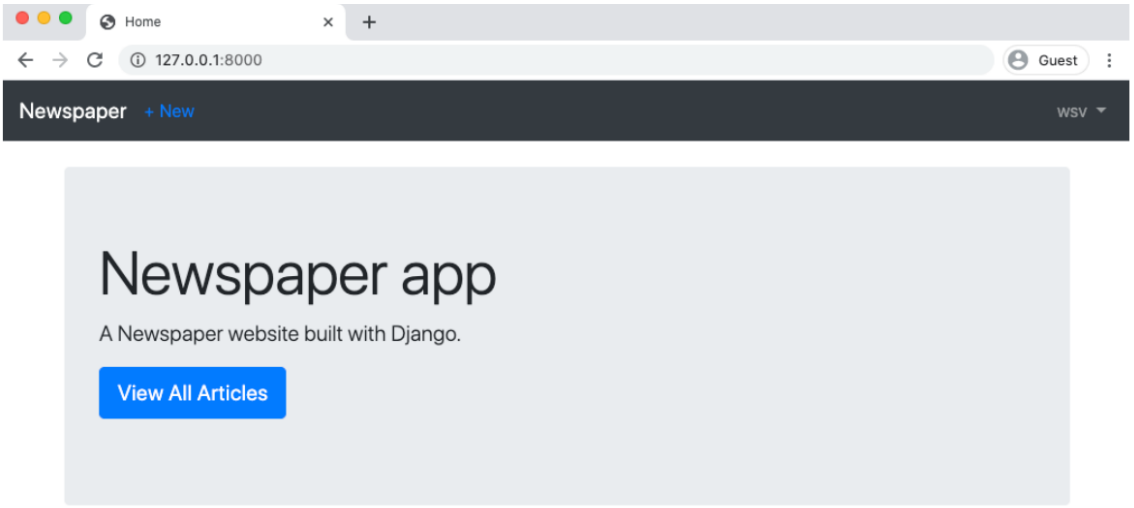
لماذا لا تستخدم Bootstrap لتحسين صفحتنا الرئيسية الأصلية الآن أيضًا؟ يمكننا تحديث نموذج home.html على النحو التالي.

Code

```
<!-- templates/home.html -->
{% extends 'base.html' %}
{% block title %}Home{% endblock title %}
{% block content %}
<br/>
<div class="jumbotron">
<h1 class="display-4">Newspaper app</h1>
<p class="lead">A Newspaper website built with Django.</p>
<p class="lead">
<a class="btn btn-primary btn-lg" href="{% url 'article_list' %}"
role="button">View All Articles</a>
</p>
</div>
{% endblock content %}
```

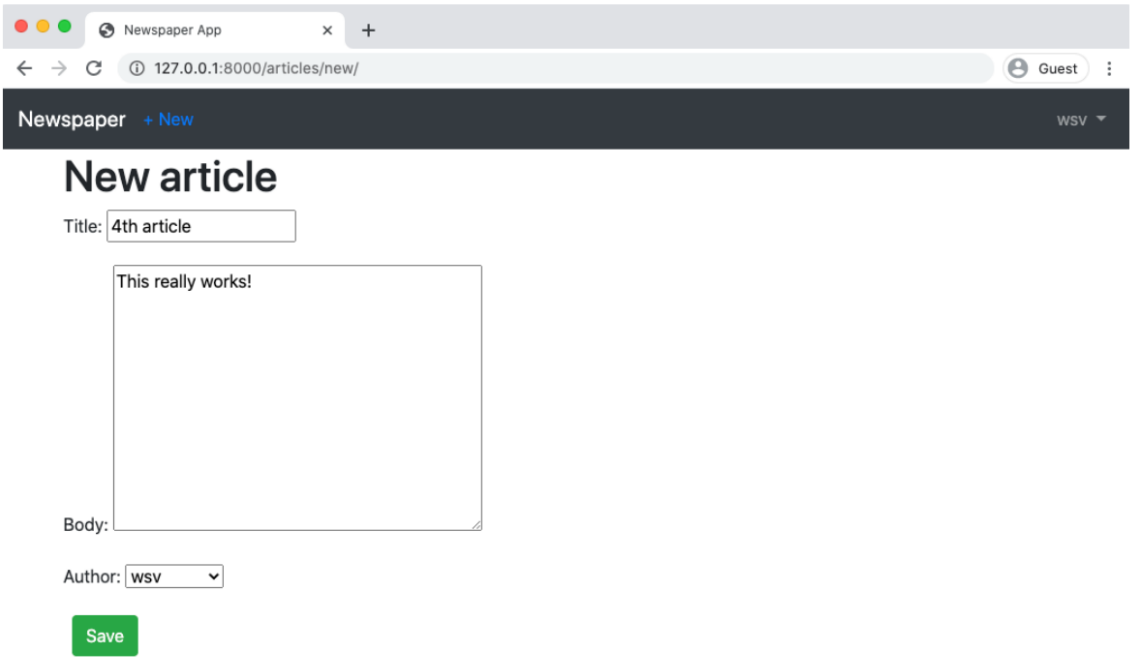
لقد انتهينا جميعا. دعنا فقط نؤكد أن كل شيء يعمل كما هو متوقع. انتقل إلى صفحتنا الرئيسية على

http://127.0.0.1:8000/



Homepage with new link in nav

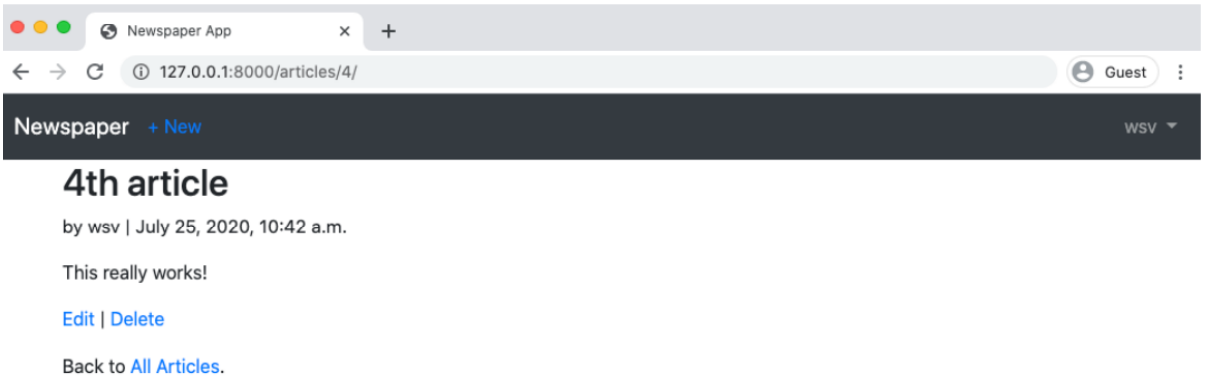
Click on the link for "+ New" in the top navbar and you'll be redirected to our create page.



Create page

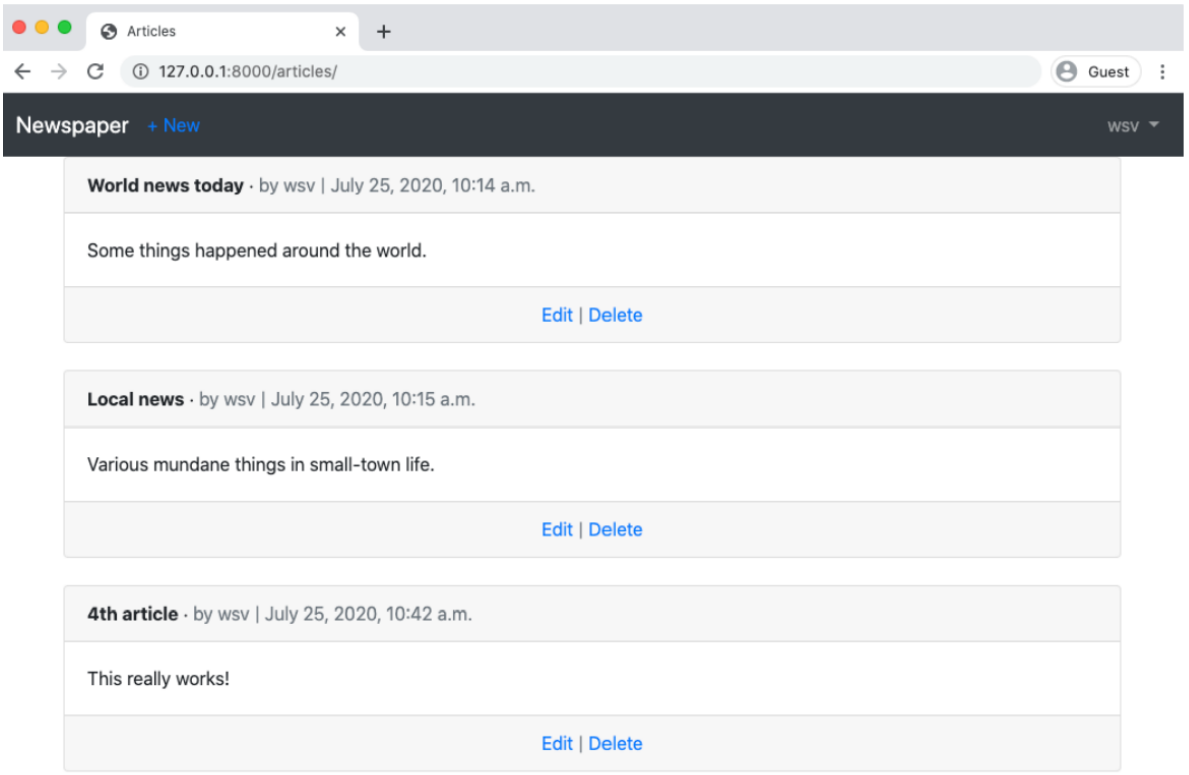
المضي قدما وإنشاء مقال جديد. ثم انقر فوق الزر "حفظ". ستتم إعادة توجيهك إلى صفحة التفاصيل. لماذا ا؟ لأنه في ملف Models.py الخاص بنا ، قمنا بتعيين طريقة get_absolute_url على article_detail. هذه طريقة جيدة لأننا إذا قمنا لاحقًا بتغيير نمط عنوان url لصفحة التفاصيل إلى ، على سبيل المثال ، مقالات / تفاصيل / 4 / ، فستظل

عملية إعادة التوجيه تعمل. أيًا كان المسار المرتبط بـ article_detail ، فسيتم استخدامه. لا يوجد تشفير ثابت للطريق نفسه



Detail page

لاحظ أيضًا أن المفتاح الأساسي هنا هو 4 في عنوان URL. على الرغم من أننا لا نعرض سوى ثلاثة مقالات في الوقت الحالي ، فإن Django لا يعيد ترتيب المفاتيح الأساسية لمجرد أننا حذفنا واحدة. من الناحية العملية ، لا تحذف معظم مواقع العالم الحقيقي أي شيء فعليًا ؛ بدلاً من ذلك ، يقومون "بإخفاء" الحقول المحذوفة لأن هذا يسهل الحفاظ على تكامل قاعدة البيانات ويعطي خيار "إلغاء الحذف" لاحقًا إذا لزم الأمر. من خلال نهجنا الحالي بمجرد حذف شيء ما ، فإنه يختفي نهائيًا! انقر على رابط "كل المقالات" لرؤية صفحة المقالات الجديدة



Updated articles page

هناك مقالنا الجديد في الأسفل كما هو متوقع. الخلاصة لقد أنشأنا تطبيق مقالات مخصصًا بوظيفة CRUD. لكن لا توجد أذونات أو تصاريح حتى الآن ، مما يعني أنه يمكن لأي شخص فعل أي شيء! يمكن للمستخدم الذي سجّل الخروج زيارة جميع عناوين URL ويمكن لأي مستخدم سجّل الدخول إجراء تعديلات أو حذف لمقالة موجودة ، حتى لو لم تكن تخصه! في الفصل التالي سنضيف الأذونات والتراخيص إلى مشروعنا لإصلاح ذلك.