Chapter 15: Comments

هناك طريقتان يمكننا من خلالهما إضافة تعليقات إلى موقع الصحف الخاص بنا. الأول هو إنشاء تطبيق مخصص للتعليقات وربطه بالمقالات ، ولكن يبدو أن هذا يبدو وكأنه مبالغة في الهندسة في هذه المرحلة. بدلاً من ذلك ، يمكننا ببساطة إضافة نموذج إضافي يسمى التعليق إلى تطبيق المقالات الخاص بنا وربطه بنموذج المقالة من خلال مفتاح خارجي. سنتخذ نهجًا أبسط لأنه من السهل دائمًا إضافة المزيد من التعقيد لاحقًا. بحلول نهاية هذا الفصل ، سيكون لدى المستخدمين القدرة على ترك تعليقات على أي مقالات مستخدمين آخرين. نموذج للبدء ، يمكننا إضافة جدول آخر إلى قاعدة البيانات الحالية لدينا يسمى التعليق. سيكون لهذا النموذج علاقة مفتاح خارجي متعدد الأطراف بالمقال: يمكن أن تحتوي مقالة واحدة على العديد من التعليقات ، لكن ليس العكس. تقليديًا ، يكون اسم حقل المفتاح الخارجي هو ببساطة النموذج الذي يرتبط به ، لذلك سيُطلق على هذا الحقل اسم المقالة. المجالان الآخران سيكونان التعليق والمؤلف. افتح ملف articles / Models.py وأسفل الكود الحالي أضف ما يلي.

```
class Comment(models.Model): # new
    article = models.ForeignKey(Article, on_delete=models.CASCADE)
    comment = models.CharField(max_length=140)
    author =
    models.ForeignKey(get_user_model(),on_delete=models.CASCADE,)
    def __str__(self):
        return self.comment
  def get_absolute_url(self):
        return reverse('article_list')
```

يحتوي نموذج التعليق أيضًا على طريقة __str__ وطريقة get_absolute_url تعود إلى المقالات / الصفحة الرئيسية. نظرًا لأننا حدّثنا نماذجنا ، فقد حان الوقت لإنشاء ملف ترحيل جديد ثم تطبيقه. لاحظ أنه من خلال إضافة مقالات في نهاية الأمر makemigrations - وهو أمر اختياري - فإننا نحدد أننا نريد استخدام تطبيق Articles فقط هنا. هذه عادة جيدة للاستخدام. على سبيل المثال ، ماذا لو قمنا بإجراء تغييرات على النماذج في تطبيقين مختلفين؟ إذا لم نحدد تطبيقًا ، فسيتم دمج تغييرات كلا التطبيقين في ملف الترحيل نفسه مما يجعل تصحيح الأخطاء في المستقبل أكثر صعوبة. اجعل كل عملية ترحيل صغيرة ومحتواه قدر الإمكان.

Command Line

(news) $ python manage.py makemigrations articles

(news) $ python manage.py migrate

Admin

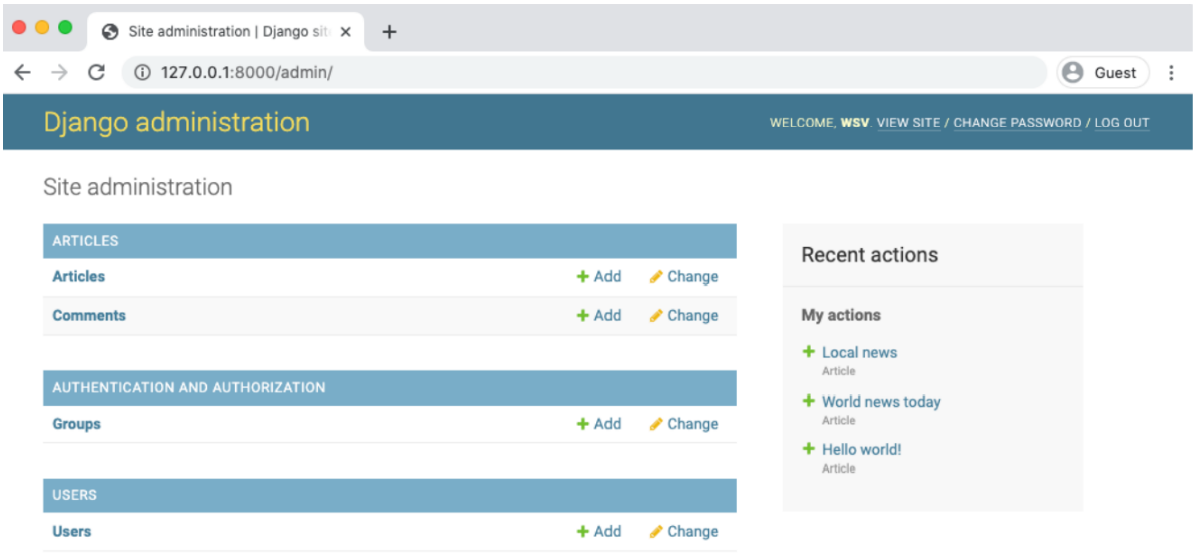
بعد إنشاء نموذج جديد ، من الجيد التلاعب به في تطبيق المسؤول قبل عرضه على موقعنا الإلكتروني الفعلي. أضف تعليقًا إلى ملف admin.py الخاص بنا حتى يكون مرئيًا.

Code

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

ثم ابدأ تشغيل الخادم باستخدام python manager.py runserver وانتقل إلى صفحتنا الرئيسية

http://127.0.0.1:8000/admin/



تحت "المقالات" التطبيق الخاص بنا ، سترى جدولين لدينا: التعليقات والمقالات. انقر فوق "+ إضافة" بجوار التعليقات. سترى أنه ضمن "المقالة" توجد قائمة منسدلة بالمقالات الحالية ، نفس الشيء للمؤلف ، وهناك حقل نص بجوار "التعليق".

حدد مقالة ، واكتب تعليقًا ، ثم حدد مؤلفًا ليس admin لديك ، وربما المختبِر كما فعلت في الصورة. ثم انقر فوق الزر "حفظ" , يجب أن ترى بعد ذلك تعليقك في صفحة "التعليقات"

في هذه المرحلة ، يمكننا إضافة حقل مشرف إضافي حتى نتمكن من رؤية التعليق والمقال في هذه الصفحة. ولكن أليس من الأفضل رؤية جميع نماذج التعليقات المتعلقة بنموذج Post واحد فقط؟ اتضح أنه يمكننا ذلك باستخدام ميزة إدارة Django المسماة inlines والتي تعرض علاقات المفاتيح الخارجية بطريقة مرئية لطيفة. هناك نوعان من طرق العرض المضمنة الرئيسية المستخدمة: TabularInline و StackedInline. الاختلاف الوحيد بينهما هو نموذج عرض المعلومات. في جدول ، تظهر جميع حقول النموذج في سطر واحد بينما في StackedInline ، يكون لكل حقل سطر خاص به. سننفذ كلا الأمرين حتى تتمكن من تحديد الخيار الذي تفضله. قم بتحديث المقالات / admin.py على النحو التالي في محرر النصوص الخاص بك

Code

```
# articles/admin.py
from django.contrib import admin
from .models import Article, Comment
class CommentInline(admin.StackedInline): # new
        model = Comment
class ArticleAdmin(admin.ModelAdmin): # new
        inlines = [
        CommentInline,
        ]
admin.site.register(Article, ArticleAdmin) # new
admin.site.register(Comment)
```

عد الآن إلى صفحة الإدارة الرئيسية في

http://127.0.0.1:8000/admin/

وانقر على "المقالات". حدد المقالة التي أضفت تعليقًا لها للتو والتي كانت "المقالة الرابعة" في حالتي.

أفضل ، أليس كذلك؟ يمكننا رؤية جميع المقالات والتعليقات ذات الصلة وتعديلها في مكان واحد. لاحظ أنه افتراضيًا ، سيعرض مسؤول Django هنا 3 صفوف فارغة. يمكنك تغيير الرقم الافتراضي الذي يظهر مع الحقل الإضافي. لذلك إذا لم تكن تريد حقولًا بشكل افتراضي ، فسيبدو الرمز كما يلي:

Code

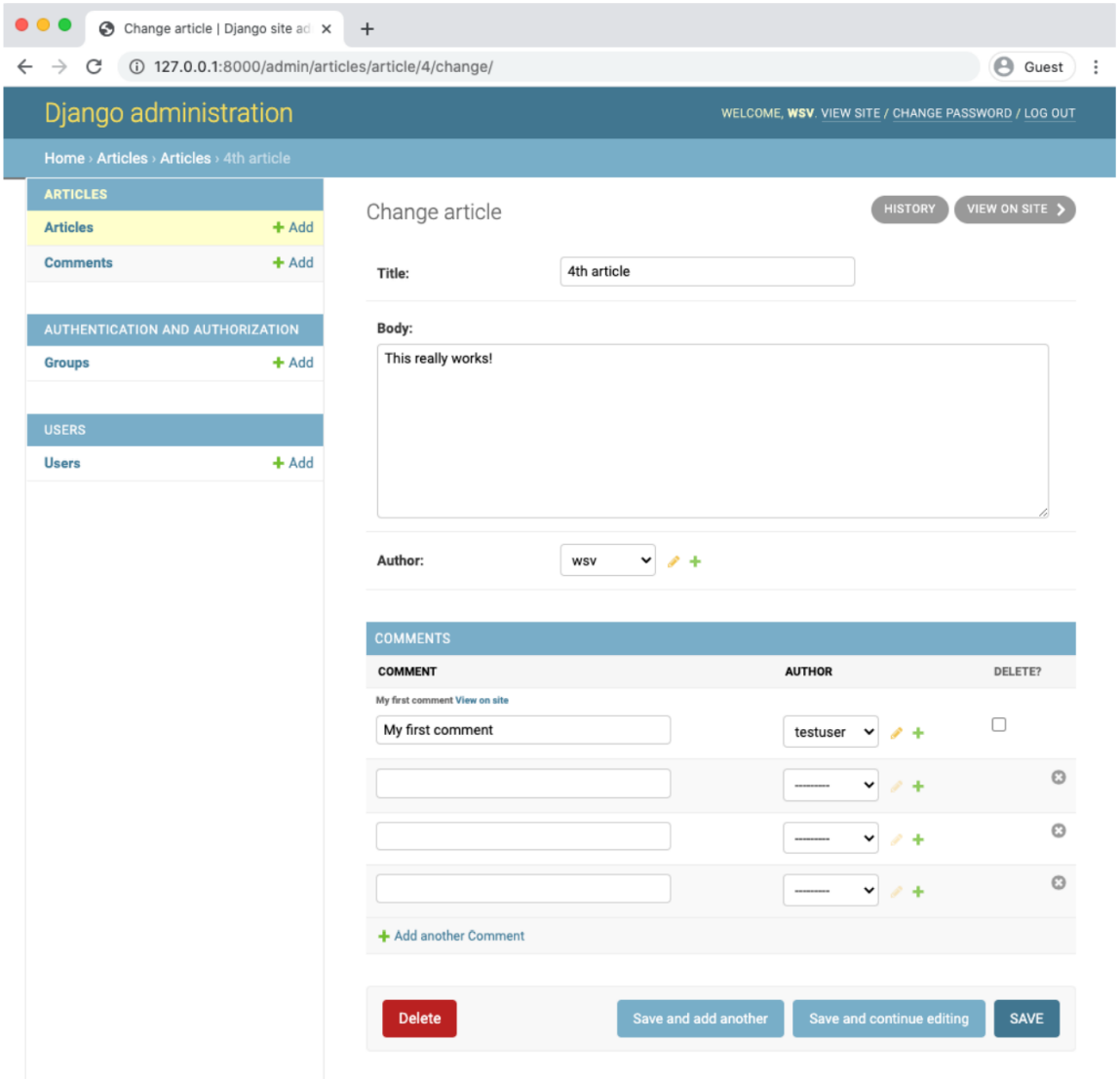
```
# articles/admin.py
...
class CommentInline(admin.StackedInline):
        model = Comment
        extra = 0 # new
```

أنا شخصياً ، على الرغم من ذلك ، أفضل استخدام TabularInline لأنه يعرض معلومات أكثر في مساحة أقل. للتبديل إليه ، نحتاج فقط إلى تغيير CommentInline من admin.StackedInline إلى admin.TabularInline.

Code

```
# articles/admin.py
from django.contrib import admin
from .models import Article, Comment
class CommentInline(admin.TabularInline): # new
        model = Comment
class ArticleAdmin(admin.ModelAdmin):
        inlines = [
        CommentInline,
        ]
admin.site.register(Article, ArticleAdmin)
admin.site.register(Comment)
```

قم بتحديث صفحة المسؤول وسترى التغيير الجديد: يتم عرض جميع الحقول لكل نموذج في نفس السطر.



أفضل بكثير. الآن نحن بحاجة إلى تحديث القالب الخاص بنا لعرض التعليقات. نموذج منذ التعليق موجود داخل تطبيق المقالات الحالي لدينا فقط نحتاج إلى تحديث القوالب الحالية لـ article_list.html و article_detail.html لعرض المحتوى الجديد الخاص بنا. لا يتعين علينا إنشاء نماذج جديدة والتلاعب بعناوين URL وطرق العرض. ما نريد القيام به هو عرض جميع التعليقات المتعلقة بمقال معين. هذا يسمى "استعلام" لأننا نطلب قاعدة البيانات للحصول على جزء محدد من المعلومات. في حالتنا ، بالعمل باستخدام مفتاح خارجي ، نريد أن نتبع علاقة إلى الوراء 177: لكل مقالة ابحث عن نماذج التعليقات ذات الصلة. يحتوي Django على بناء جملة مدمج للعلاقات التالية "للخلف" 178 المعروفة باسم FOO_set حيث FOO هو اسم نموذج المصدر ذي الأحرف المنخفضة. لذلك بالنسبة إلى نموذج المقالة الخاص بنا ، يمكننا استخدام article_set للوصول إلى جميع مثيلات النموذج. لكنني شخصياً لا أحب هذا النحو بشدة حيث أجده مربكًا وغير بديهي. تتمثل الطريقة الأفضل في إضافة سمة ذات صلة بالاسم إلى نموذجنا مما يتيح لنا تعيين اسم

هذه العلاقة العكسية بشكل صريح بدلاً من ذلك. لنفعل ذلك. للبدء ، أضف سمة ذات صلة بالاسم إلى نموذج التعليق الخاص بنا. الافتراضي الجيد هو تسمية صيغة الجمع للنموذج الذي يحمل المفتاح الأجنبي.

Code

```
# articles/models.py
...
class Comment(models.Model):
    article =
    models.ForeignKey(Article,on_delete=models.CASCADE,related_name='com
    ments', )
    comment = models.CharField(max_length=140)
    author =
    models.ForeignKey(get_user_model(),on_delete=models.CASCADE,)
    def __str__(self):
        return self.comment
    def get_absolute_url(self):
        return reverse('article_list')
```

نظرًا لأننا أجرينا تغييرًا على نموذج قاعدة البيانات لدينا ، نحتاج إلى إنشاء ملف تهجير وتحديث قاعدة البيانات. أوقف الخادم المحلي باستخدام Control + c وقم بتنفيذ الأمرين التاليين. ثم قم بتدوير الخادم مرة أخرى حيث سنستخدمه قريبًا.

```
Command Line
(news) $ python manage.py makemigrations articles
Migrations for 'articles':
articles/migrations/0003_auto_20200726_1405.py
- Alter field article on comment
(news) $ python manage.py migrate
Operations to perform:
Apply all migrations: admin, articles, auth, contenttypes,
sessions, users
Running migrations:
Applying articles.0003_auto_20200726_1405... OK
```

```
(news) $ python manage.py runserver
```

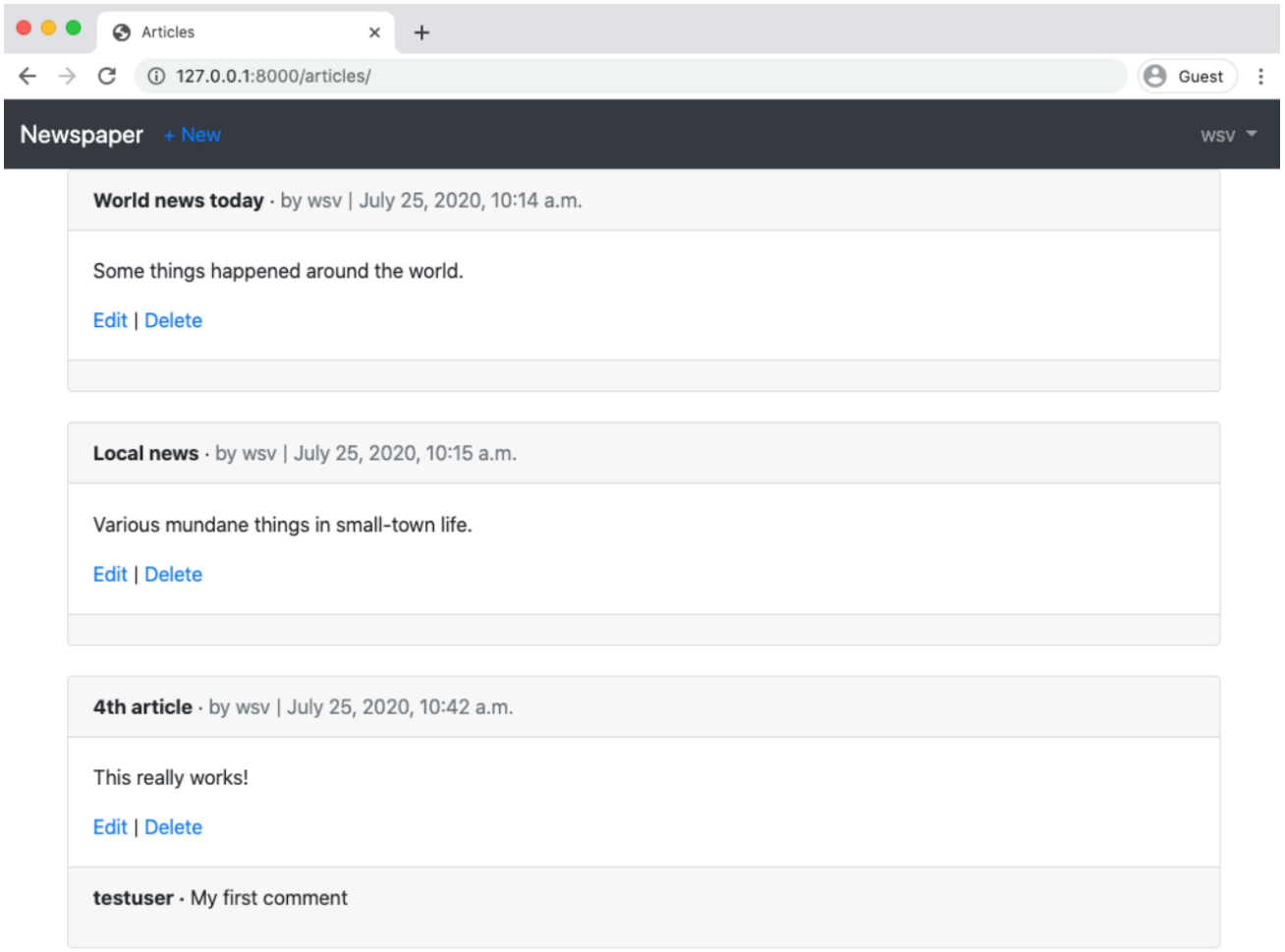
يستغرق فهم الاستعلامات بعض الوقت ، لذا لا تقلق إذا كانت فكرة العلاقات العكسية محيرة. سأوضح لك كيفية تنفيذ الشفرة حسب الرغبة. وبمجرد أن تتقن هذه الحالات الأساسية ، يمكنك استكشاف كيفية تصفية مجموعات طلبات البحث الخاصة بك بتفاصيل رائعة بحيث تعرض بالضبط المعلومات التي تريدها. في ملف article_list.html الخاص بنا ، يمكننا إضافة تعليقاتنا إلى تذييل البطاقة. لاحظ أنني نقلت روابط التعديل والحذف إلى نص البطاقة. للوصول إلى كل تعليق ، نطلق على article.comments.all مما يعني أولاً إلقاء نظرة على نموذج المقالة ، ثم التعليقات وهو الاسم المرتبط بنموذج التعليق بالكامل ، وتحديد كل ما تم تضمينه. قد يستغرق الأمر بعض الوقت حتى تعتاد على بناء الجملة هذا للإشارة إلى بيانات المفتاح الخارجي في قالب!

Code

```
<!-- template/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">
<!-- Changes start here! -->
<p>{{ article.body }}</p>
<a href="{% url 'article_edit' article.pk %}">Edit</a> |
<a href="{% url 'article_delete' article.pk %}">Delete</a>
</div>
```

```
<div class="card-footer">
{% for comment in article.comments.all %}
<p>
<span class="font-weight-bold">
{{ comment.author }} &middot;
</span>
{{ comment }}
</p>
{% endfor %}
</div>
<!-- Changes end here! -->
</div>
<br />
{% endfor %}
{% endblock content %}
```

إذا قمت بتحديث صفحة المقالات على http://127.0.0.1:8000/articles/ يمكننا أن نرى تعليقنا الجديد معروضًا على الصفحة



يوهوو! إنها تعمل. يمكننا أن نرى التعليقات مدرجة أسفل الرسالة الأولية. مع مزيد من الوقت ، سنركز على النماذج الآن حتى يتمكن المستخدم من كتابة مقال جديد مباشرة على المقالات / الصفحة ، بالإضافة إلى إضافة تعليقات أيضًا. لكن التركيز الرئيسي لهذا الفصل هو توضيح كيفية عمل العلاقات الخارجية الرئيسية في Django. الخلاصة تطبيق الصحف الخاص بنا اكتمل الآن. لديه تدفق قوي لمصادقة المستخدم يستخدم نموذج مستخدم مخصصًا والبريد الإلكتروني. تحسين التصميم بفضل Bootstrap. وكلا من المقالات والتعليقات. حتى أننا وضعنا أصابع قدمنا في الأذونات والتراخيص. مهمتنا المتبقية هي نشرها على الإنترنت. لقد ازدادت عملية النشر تعقيدًا مع كل تطبيق تالٍ ، لكننا ما زلنا نأخذ اختصارات حول الأمان والأداء. في الفصل التالي ، سنرى كيفية نشر موقع Django بشكل صحيح باستخدام متغيرات البيئة و PostgreSQL وإعدادات إضافية