

Chapter 14: Permissions and Authorization

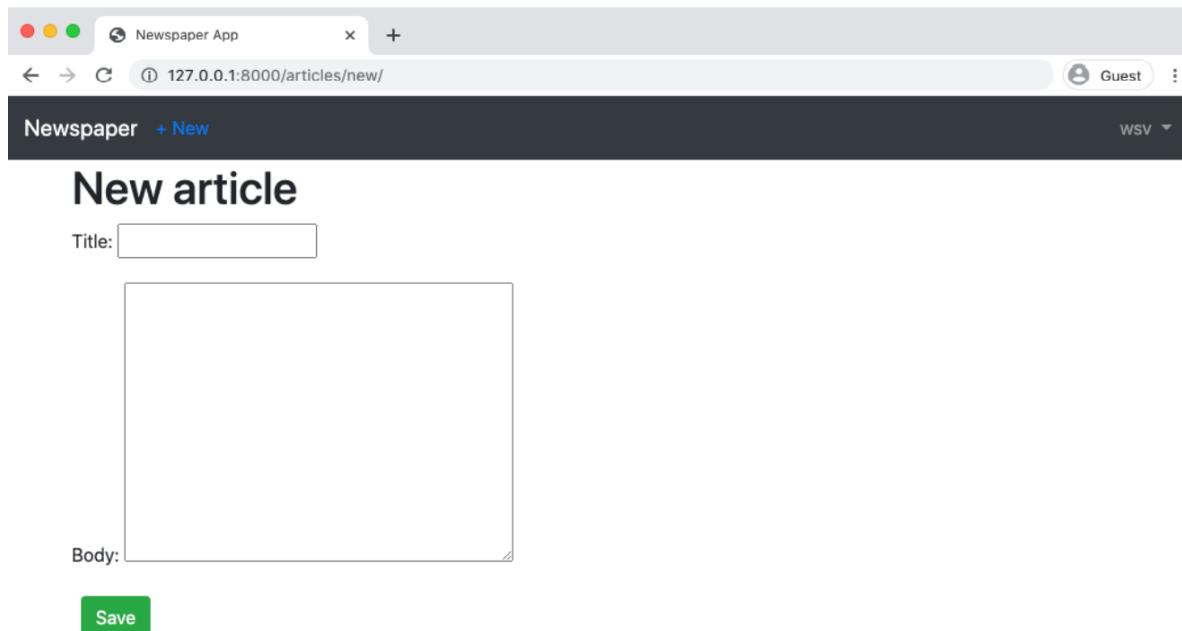
هناك العديد من المشكلات في موقع الصحف الحالي الخاص بنا. لسبب واحد ، نريد أن تكون جريدتنا مستدامة مالياً. مع مزيد من الوقت ، يمكننا إضافة تطبيق مدفوعات لتحصيل رسوم للوصول ، ولكن في الوقت الحالي سنطلب من المستخدم تسجيل الدخول لعرض أي مقالات. يُعرف هذا باسم التفويض. من الشائع تعيين قواعد مختلفة حول الشخص المخول له عرض مناطق من موقعك. لاحظ أن هذا يختلف عن المصادقة وهي عملية تسجيل المستخدمين وتسجيل دخولهم. الإذن يقيد الوصول ؛ المصادقة تمكن المستخدم من التسجيل وتدفع تسجيل الدخول. كإطار عمل ويب ناضج ، فإن Django لديه وظائف مضمنة للترخيص يمكننا استخدامها بسرعة. في هذا الفصل ، سنقتصر الوصول إلى الصفحات المختلفة على المستخدمين المسجلين فقط. تحسين CreateView في الوقت الحاضر يمكن تعيين المؤلف في مقال جديد إلى أي مستخدم. وبدلاً من ذلك ، يجب ضبطه تلقائياً على المستخدم الحالي. يوفر برنامج CreateView الافتراضي الكثير من الوظائف لنا ، ولكن من أجل تعيين المستخدم الحالي على المؤلف ، نحتاج إلى تخصيصه. سنقوم بإزالة المؤلف من الحقول وبدلاً من ذلك نقوم بتعيينه تلقائياً عبر الطريقة form_valid.

Code

```
# articles/views.py
...
class ArticleCreateView(CreateView):
    model = Article
    template_name = 'article_new.html'
    fields = ('title', 'body') # new
    def form_valid(self, form): # new
        form.instance.author = self.request.user
        return super().form_valid(form)
...
```

كيف عرفت أنه يمكنني تحديث CreateView بهذا الشكل؟ الجواب هو أنني نظرت إلى شفرة المصدر واستخدمت Google. تعد العروض العامة المستندة إلى الفصل رائعة لبدء مشاريع جديدة ولكن عندما تريد تخصيصها ، فمن الضروري أن تشمر عن سواعدك وتبدأ في فهم ما يجري تحت الغطاء. كلما زاد استخدامك

لطرق العرض المضمنة وتخصيصها ، ستصبح أكثر راحة عند إجراء التخصيصات مثل هذا. الآن أعد تحميل المتصفح وحاول النقر على رابط "+" جديد" في التنقل العلوي. سيعيد التوجيه إلى صفحة الإنشاء المحدثة حيث لم يعد المؤلف حقلًا.



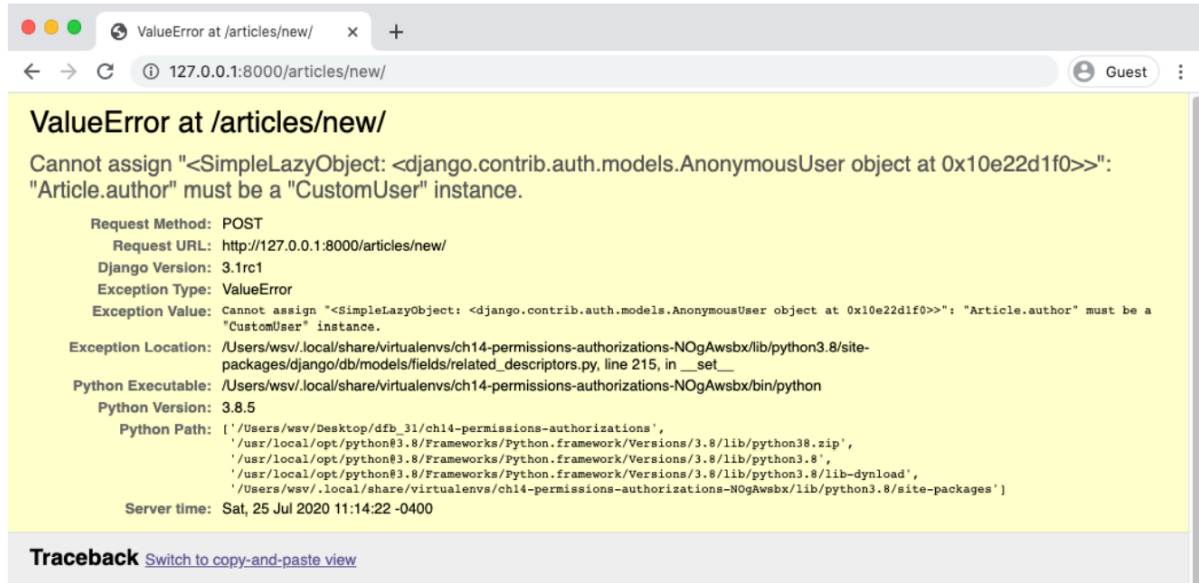
The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/articles/new/'. The page title is 'Newspaper + New'. The main heading is 'New article'. Below the heading, there is a 'Title:' label followed by a text input field. Below the title field is a large text area for the 'Body:'. At the bottom left of the body area is a green 'Save' button.

New article link

إذا قمت بإنشاء مقال جديد ثم انتقلت إلى المسؤول ، فسترى أنه قد تم ضبطه تلقائيًا على المستخدم الحالي الذي قام بتسجيل الدخول. التصاريح هناك العديد من المشكلات حول عدم وجود التراخيص في مشروعنا الحالي. من الواضح أننا نود تقييد الوصول إلى المستخدمين فقط ، لذلك لدينا خيار شحن القراء يومًا ما إلى جريدتنا. ولكن بعد ذلك ، يمكن لأي مستخدم قام بتسجيل الخروج عشوائيًا يعرف عنوان URL الصحيح يمكنه الوصول إلى أي جزء من الموقع. ضع في اعتبارك ما سيحدث إذا حاول مستخدم سجل الخروج إنشاء مقال جديد؟ لتجربتها ، انقر فوق اسم المستخدم الخاص بك في الزاوية اليمنى العليا من شريط التنقل ، ثم حدد "تسجيل الخروج" من خيارات القائمة المنسدلة. يختفي الرابط "+" جديد" من شريط التنقل ولكن ماذا يحدث إذا انتقلت إليه مباشرة؟ الصفحة لا تزال هناك ؟

<http://127.0.0.1:8000/articles/new/>

حاول الآن إنشاء مقال جديد بعنوان وجسم. انقر فوق الزر "حفظ".



Create page error

خطا! هذا لأن نموذجنا يتوقع حقل مؤلف مرتبط بالمستخدم الحالي الذي قام بتسجيل الدخول. ولكن نظرًا لأننا لم نسجل الدخول ، فلا يوجد مؤلف ، وبالتالي فشل الإرسال. ما يجب القيام به؟ Mixins من الواضح أننا نريد تعيين بعض التراخيص بحيث لا يتمكن سوى المستخدمين الذين قاموا بتسجيل الدخول من الوصول إلى الموقع. للقيام بذلك ، يمكننا استخدام mixin ، وهو نوع خاص من الوراثة المتعددة التي يستخدمها Django لتجنب الكود المكرر ولا يزال يسمح بالتخصيص. على سبيل المثال ، يحتاج ListView168 العام المدمج إلى طريقة لإرجاع قالب. ولكن الأمر كذلك مع DetailView169 وفي الواقع كل طريقة عرض أخرى تقريبًا. بدلاً من تكرار نفس الكود في كل عرض عام كبير ، يقسم Django هذه الوظيفة إلى "mixin" يُعرف باسم TemplateResponseMixin170. يستخدم كل من ListView و DetailView هذا المزيج لتقديم القالب المناسب. إذا قرأت شفرة مصدر Django ، والمتاحة مجانًا على Github171 ، فسترى مزيجًا مستخدمًا في كل مكان. لتقييد الوصول إلى العرض على المستخدمين المسجلين فقط ، لدى Django ملف تسجيل الدخول المطلوب mixin172 الذي يمكننا استخدامه. إنه قوي وموجز للغاية. في ملف articles / views.py ، قم باستيراد LoginRequiredMixin في الأعلى ثم قم بإضافته إلى articleCreateView. تأكد من أن mixin على يسار CreateView حتى تتم قراءته أولاً. نريد أن يعلم برنامجنا CreateView بالفعل أننا نعتزم تقييد الوصول. وهذا كل شيء! لقد انتهينا.

Code

```
# articles/views.py
```

```
from django.contrib.auth.mixins import LoginRequiredMixin # new
```

```

from django.views.generic import ListView, DetailView
from django.views.generic.edit import UpdateView, DeleteView, CreateView
from django.urls import reverse_lazy
from .models import Article
...
class ArticleCreateView(LoginRequiredMixin, CreateView): # new
...

```

عد الآن إلى الصفحة الرئيسية باختصار في `http://127.0.0.1:8000/` حتى نتجنب إعادة إرسال النموذج. ثم انتقل إلى عنوان URL للرسالة الجديدة مباشرة مرة أخرى على

<http://127.0.0.1:8000/articles/new/>.

سترى الخطأ التالي "لم يتم العثور على الصفحة":



Error page

ماذا يحدث؟ أعاد Django توجيه المستخدمين تلقائيًا إلى صفحة تسجيل الدخول ، تمامًا كما نرغب! LoginRequiredMixin الآن نرى أن تقييد الوصول إلى العرض يتطلب إضافة LoginRequiredMixin في بداية جميع طرق العرض الحالية وتحديد login_url الصحيح. دعنا نحدّث بقية طرق عرض المقالات لأننا لا نريد أن يتمكن المستخدم من إنشاء رسالة أو قراءتها أو تحديثها أو حذفها إذا لم يتم تسجيل الدخول. يجب أن يبدو ملف `views.py` الكامل الآن على النحو التالي:

Code

```

# articles/views.py

from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic import ListView, DetailView

```

```

from django.views.generic.edit import CreateView, UpdateView, DeleteView
from django.urls import reverse_lazy
from .models import Article

class ArticleListView(LoginRequiredMixin, ListView): # new
    model = Article
    template_name = 'article_list.html'

class ArticleDetailView(LoginRequiredMixin, DetailView): # new
    model = Article
    template_name = 'article_detail.html'

class ArticleUpdateView(LoginRequiredMixin, UpdateView): # new
    model = Article
    fields = ('title', 'body',)
    template_name = 'article_edit.html'

class ArticleDeleteView(LoginRequiredMixin, DeleteView): # new
    model = Article
    template_name = 'article_delete.html'
    success_url = reverse_lazy('article_list')

class ArticleCreateView(LoginRequiredMixin, CreateView):
    model = Article
    template_name = 'article_new.html'
    fields = ('title', 'body',)
    def form_valid(self, form):
        form.instance.author = self.request.user
        return super().form_valid(form)

```

انطلق والعب مع الموقع للتأكد من أن عمليات إعادة توجيه تسجيل الدخول تعمل الآن كما هو متوقع. إذا كنت

بحاجة إلى مساعدة في تذكر ماهية عناوين URL المناسبة ، فقم بتسجيل الدخول أولاً وقم بتدوين عناوين

URL لكل مسار من مسارات الإنشاء والتعديل والحذف وجميع المقالات. `DeleteView` و `UpdateView`

نحن نحرص تقدماً ولكن لا تزال هناك مشكلة تتعلق بالتعديل وحذف طرق العرض. يمكن لأي مستخدم قام بتسجيل الدخول إجراء تغييرات على أي مقال. ما نريده هو تقييد هذا الوصول بحيث لا يحصل هذا الإذن إلا على مؤلف المقال. يمكننا إضافة منطق أدونات لكل عرض لهذا ولكن الحل الأكثر أناقة هو إنشاء مزيج مخصص ، فئة ذات ميزة معينة نريد إعادة استخدامها في كود Django الخاص بنا. والأفضل من ذلك ، فإن Django يشحن مع mixin المدمج ، UserPassesTestMixin173 ، لهذا الغرض فقط! لاستخدام UserPassesTestMixin ، قم أولاً باستيراده في الجزء العلوي من ملف articles / views.py ثم قم بإضافته إلى كل من طرق العرض الخاصة بالتحديث والحذف حيث نريد هذا التقييد. يتم استخدام طريقة test_func بواسطة UserPassesTestMixin لمنطقنا. نحن بحاجة إلى تجاوزه. في هذه الحالة ، قمنا بتعيين المتغير obj على الكائن الحالي الذي تم إرجاعه بواسطة طريقة العرض باستخدام get_object(). ثم نقول ، إذا كان المؤلف الموجود على الكائن الحالي يطابق المستخدم الحالي على صفحة الويب (من قام بتسجيل الدخول ويحاول إجراء التغيير) ، فقم بالسماح بذلك. إذا كان خطأ ، فسيتم إلقاء خطأ تلقائياً. يبدو الرمز كما يلي:

Code

```
# articles/views.py

from django.contrib.auth.mixins import (
    LoginRequiredMixin,
    UserPassesTestMixin # new
)

from django.views.generic import ListView, DetailView
from django.views.generic.edit import UpdateView, DeleteView, CreateView
from django.urls import reverse_lazy
from .models import Article
...

class ArticleUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView): #
    new

    model = Article

    fields = ('title', 'body',)

    template_name = 'article_edit.html'
```

```

def test_func(self): # new
    obj = self.get_object()
    return obj.author == self.request.user

class ArticleDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView): #
new
    model = Article
    template_name = 'article_delete.html'
    success_url = reverse_lazy('article_list')
    def test_func(self): # new
        obj = self.get_object()
        return obj.author == self.request.user

```

الآن قم بتسجيل الخروج من حساب المستخدم المتميز الخاص بك وقم بتسجيل الدخول باستخدام `testuser`. إذا كان الكود يعمل ، فلا يجب أن تكون قادرًا على تعديل أو حذف أي منشورات كتبها مستخدمك المتميز ، وكلها الآن. بدلاً من ذلك سترى صفحة خطأ "تم رفض الإذن 403".



خاتمة تطبيق الصحف لدينا على وشك الانتهاء. هناك خطوات أخرى يمكن أن نتخذها في هذه المرحلة ، مثل عرض روابط التحرير والحذف للمستخدمين المناسبين فقط ، والتي قد تتضمن علامات قوالب مخصصة 174 ولكن بشكل عام التطبيق في حالة جيدة. لقد تم تكوين مقالاتنا بشكل صحيح ، وضبط الأذونات والتراخيص ، ومصادقة المستخدم بالترتيب. العنصر الأخير المطلوب هو قدرة زملائك المستخدمين المسجلين على ترك التعليقات التي سنغطيها في الفصل التالي.