Práctica Opcional #3: Crea un modelo de comentarios

Actualmente, solamente tenemos un modelo de Post, ¿Qué si queremos recibir retro alimentación de tus lectores y dejarlos comentar?

Crea un modelo de comentarios para el blog

Vamos a abrir `blog/models.py` y pega esta pieza de código al final del archivo:

```
class Comment(models.Model):
    post = models.ForeignKey('blog.Post', related_name='comments')
    author = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(default=timezone.now)
    approved_comment = models.BooleanField(default=False)

    def approve(self):
        self.approved_comment = True
        self.save()

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

Puedes ir atrás a la práctica de Modelos en Django si necesitas refrescar lo que cada tipo de campo significa.

En esta práctica vamos a tener un nuevo tipo de campo:

- `models.BooleanField` - Es un campo cierto/falso.

La opción `related_name` en `models.ForeignKey` nos permite tener acceso a los comentarios desde el modelo Post.

Creando tablas para los modelos en tu base de datos.

Es tiempo de agregar nuestro modelo de comentarios a la base de datos. Para hacer esto le vamos a decir a Django que haga los cambios a nuestro modelo. Escribe `python manage.py makemigrations blog` en tu línea de comandos. Deberías ver algo como esto:

```
(myvenv) ~/django-daw$ python manage.py makemigrations blog
Migrations for 'blog':
  0002_comment.py:
    - Create model Comment
```

Como puedes ver, se ha creado otro archivo de migración en la carpeta `blog/migrations/`. Ahora necesitamos aplicar estos cambios escribiendo `python manage.py migrate blog` en la línea de comandos. La salida debería verse así:

```
(myvenv) ~/django-daw$ python manage.py migrate blog
Operations to perform:
  Apply all migrations: blog
Running migrations:
  Rendering model states... DONE
  Applying blog.0002_comment... OK
```

Nuestro modelo de comentarios existe ahora en la base de datos, ¿No sería genial acceder a el desde nuestro panel de administración?

Registra el modelo de comentarios en el panel de administrador

Para registrar el modelo de Comentario en el panel de administrador, ve a `blog/admin.py` y agrega esta línea:

```
admin.site.register(Comment)
```

Justo debajo de esta línea:

```
admin.site.register(Post)
```

Recuerda que es importante importar el modelo de comentarios al comienzo del archivo, se vera así:

```
from django.contrib import admin
from .models import Post, Comment

admin.site.register(Post)
admin.site.register(Comment)
```

Si escribes `python manage.py runserver` en la línea de comandos y vas a `http://127.0.0.1:8000/admin/` en tu navegador, ahora podrás acceder a la lista de

comentarios, y también tendrás la posibilidad de agregar y eliminar comentarios.

Has tus comentarios visibles

Ve al archivo `blog/templates/blog/post_detail.html` y agrega el siguiente código antes de la etiqueta `{% endblock %}`:

```
<hr>
{% for comment in post.comments.all %}
    <div class="comment">
        <div class="date">{{ comment.created_date }}</div>
        <strong>{{ comment.author }}</strong>
        <p>{{ comment.text|linebreaks }}</p>
    </div>
{% empty %}
    <p>No comments here yet :(</p>
{% endfor %}
```

Ahora puedes ver la sección de comentarios en los detalles del post.

Pero puede verse un poco mejor, así que vamos a agregar algún CSS al final del archivo `static/css/blog.css`:

```
.comment {
    margin: 20px 0px 20px 20px;
}
```

También vamos a dejar a los visitantes sobre los comentarios que dejan en la página. Ve al archivo `blog/templates/blog/post_list.html` y agrega una línea como esta:

```
<a href="{% url 'post_detail' pk=post.pk %}">Comments: {{ post.comments.count }}</a>
```

Después de esto, tu plantilla debería verse así:

```
{% extends 'blog/base.html' %}

{% block content %}
    {% for post in posts %}
        <div class="post">
            <div class="date">
                {{ post.published_date }}
            </div>
            <h1><a href="{% url 'post_detail' pk=post.pk %}">{{ post.title }}</a></h1>
            <p>{{ post.text|linebreaksbr }}</p>
            <a href="{% url 'post_detail' pk=post.pk %}">Comments: {{ post.comments.count }}</a>
        </div>
    {% endfor %}
{% endblock content %}
```

Deja a tus lectores escribir comentarios

Ahora podemos ver los comentarios hechos en nuestro blog, pero no podemos agregarlos. Cambiemos eso.

Ve a `blog/forms.py` y agrega las siguientes líneas al final del archivo:

```
class CommentForm(forms.ModelForm):

    class Meta:
        model = Comment
        fields = ('author', 'text',)
```

Recuerda importar el modelo Comentario, cambiando la línea:

```
from .models import Post
```

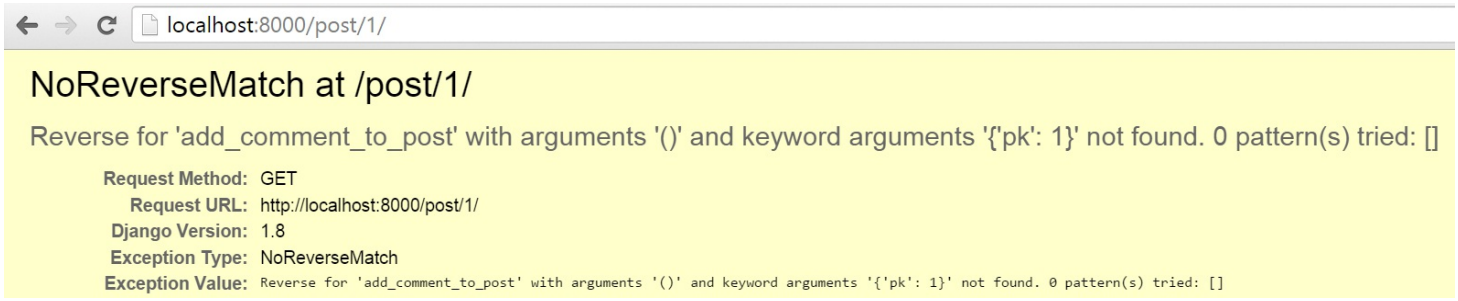
en:

```
from .models import Post, Comment
```

Ahora vamos a `blog/templates/blog/post_detail.html` y antes de la línea `{% for comment in post.comments.all %}`, agrega:

```
<a class="btn btn-default" href="{% url 'add_comment_to_post' pk=post.pk %}">Add comment</a>
```

Si quieres ir a los detalles del post, deberías ver este error:



Ahora sabemos como arreglarlo. Ve a `blog/urls.py` y agrega esto a `urlpatterns`:

```
url(r'^post/(?P<pk>\d+)/comment/$', views.add_comment_to_post, name='add_comment_to_post'),
```

Refresca la página y ahora tenemos un error diferente.

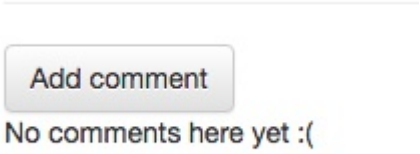
Para agregar este error, agrega esto a `blog/views.py`:

```
def add_comment_to_post(request, pk):
    post = get_object_or_404(Post, pk=pk)
    if request.method == "POST":
        form = CommentForm(request.POST)
        if form.is_valid():
            comment = form.save(commit=False)
            comment.post = post
            comment.save()
            return redirect('post_detail', pk=post.pk)
    else:
        form = CommentForm()
    return render(request, 'blog/add_comment_to_post.html', {'form': form})
```

Recuerda importar `CommentForm` al comienzo del archivo:

```
from .forms import PostForm, CommentForm
```

Ahora vamos a los detalles de la página y deberíamos ver el botón "Add Comment":



Sin embargo, cuando des click en el botón verás:

Como el error nos dice, la plantilla no existe aún. Entonces vamos a crear una en `blog/templates/blog/add_comment_to_post.html` y agregar el siguiente código:

```
{% extends 'blog/base.html' %}

{% block content %}
    <h1>New comment</h1>
    <form method="POST" class="post-form">{% csrf_token %}
        {{ form.as_p }}
        <button type="submit" class="save btn btn-default">Send</button>
    </form>
{% endblock %}
```

Ahora nuestros lectores puede agregar lo que piensan en nuestros post.

Moderando los comentarios

No todos los comentarios deberían ser mostrados. Como dueño del blog, posiblemente quieras la opción de aprobar o eliminar comentarios. Vamos a hacer algo sobre esto.

Ve a `blog/templates/blog/post_detail.html` y cambia las líneas:

```
{% for comment in post.comments.all %}
    <div class="comment">
        <div class="date">{{ comment.created_date }}</div>
        <strong>{{ comment.author }}</strong>
        <p>{{ comment.text|linebreaks }}</p>
    </div>
{% empty %}
    <p>No comments here yet :(</p>
{% endfor %}
```

a:

```
{% for comment in post.comments.all %}
    {% if user.is_authenticated or comment.approved_comment %}
    <div class="comment">
        <div class="date">
            {{ comment.created_date }}
            {% if not comment.approved_comment %}
                <a class="btn btn-default" href="{% url 'comment_remove' pk=comment.pk %}"><span class="glyphicon glyphicon-remove"></span></a>
                <a class="btn btn-default" href="{% url 'comment_approve' pk=comment.pk %}"><span class="glyphicon glyphicon-ok"></span></a>
            {% endif %}
        </div>
        <strong>{{ comment.author }}</strong>
        <p>{{ comment.text|linebreaks }}</p>
    </div>
    {% endif %}
{% empty %}
    <p>No comments here yet :(</p>
{% endfor %}
```

Deberías ver un `NoReverseMatch`, porque no hay ninguna URL que concuerde con `comment_remove` y `comment_approve` aún.

Para arreglar este error, agregamos estas URLs a `blog/urls.py`:

```
url(r'^comment/(?P<pk>\d+)/approve/$', views.comment_approve, name='comment_approve'),
url(r'^comment/(?P<pk>\d+)/remove/$', views.comment_remove, name='comment_remove'),
```

Ahora deberías ver un AttributeError. Para arreglar este error, agrega las siguiente vistas en `blog/views.py`:

```
@login_required
def comment_approve(request, pk):
    comment = get_object_or_404(Comment, pk=pk)
    comment.approve()
    return redirect('post_detail', pk=comment.post.pk)

@login_required
def comment_remove(request, pk):
    comment = get_object_or_404(Comment, pk=pk)
    comment.delete()
    return redirect('post_detail', pk=comment.post.pk)
```

Necesitas importar Comment al comienzo del archivo:

```
from .models import Post, Comment
```

Hay un pequeño cambio que podemos hacer. en nuestra página de lista – debajo de posts – actualmente vemos el número de los comentarios que el post ha recibido. Vamos a cambiar esto para ver el número de comentarios aprobados.

Para arreglar esto, ve a `blog/templates/blog/post_list.html` y cambia la línea:

```
<a href="{% url 'post_detail' pk=post.pk %}">Comments: {{ post.comments.count }}</a>
```

a:

```
<a href="{% url 'post_detail' pk=post.pk %}">Comments: {{ post.approved_comments.count }}</a>
```

finalmente, añade el método al modelo `Post` en `blog/models.py`:

```
def approved_comments(self):
    return self.comments.filter(approved_comment=True)
```

Ahora puedes ver la característica de comentarios finalizada.