Práctica Opcional #2: Asegura tu sitio

Puedes haberte dado cuenta que no usaste tu contraseña, además de cuando iniciaste en la interfaz de administrador. También debes haber notado que esto significa que cualquiera puede editar tus post en tu blog. No se tu, pero a mi no me gustaría que cualquiera editara mi blog, así que vamos a hacer algo al respecto.

Autorizando la edición y creación de post.

Primero vamos a hacer las cosas seguras. Vamos a proteger nuestras vistas `post_new`, `post_edit`, `post_draft_list`, `post_remove` y `post_publish` para que solo usuarios que hayan ingresado puedan acceder a ellas. Django nos da algunas ayudas para hacer esto, llamados *decoradores* (*decorators*). No te preocupes por las tecnicalidades ahora; puedes leer sobre eso después. El decorador que vamos a usar está en el módulo `django.contrib.auth.decorators` y es llamado `login_required`.

Entonces editemos `blog/views.py` y agrega estas líneas al comienzo del archivo junto con las demás importaciones:

```
from django.contrib.auth.decorators import login_required
```

Entonces añade la línea antes de cada una de las vistas `post_new`, `post_edit`, `post_draft_list`, `post_remove` y `post_publish`, (decorándolas) como a continuación:

```
@login_required
def post_new(request):
    [...]
```

Eso es todo. Ahora intenta acceder a `http://localhost:8000/post/new/`. ¿Notas la diferencia?

> Si obtienes un formulario vacío, posiblemente sigues logeado en la interfaz de administrador. Ve a `http://localhost:8000/admin/logout/` para salir, y luego `http://localhost:8000/post/new` de nuevo.

Puedes obtener uno de nuestros amados errores. Esto es muy interesante: El decorador que añadimos nos redireccionará a la página de ingreso, pero como no está disponible, retorna un "Page not found (404)".

No te olvides del decorador encima de `post_edit`, `post_remove`, `post_draft_list` y `post_publish` también.

Ahora otras personas no podrán crear post en nuestro blog. Desafortunadamente, nosotros tampoco podemos crearlos. Así que vamos a arreglarlo a continuación.

Autenticando usuarios

Ahora podemos intentar hacer muchas cosas mágicas para implementar usuarios y contraseñas y autenticación, pero hacer esto correctamente es complicado. Como Django viene con "baterías incluidas", alguien ya ha hecho el trabajo duro por nosotros, así que vamos a utilizarlas.

En `mysite/urls.py` agrega una URL `url(r'^accounts/login/$', views.login, name='login')`. Así el archivo debería verse similar a este:

```
from django.conf.urls import include, url
from django.contrib import admin

from django.contrib.auth import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^accounts/login/$', views.login, name='login'),
    url(r'', include('blog.urls')),
]
```

Luego necesitamos agregar una plantilla para la página de ingreso, así que crearemos el directorio `blog/templates/registration` y dentro un archivo llamado `login.html`.

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

{% block content %}
    {% if form.errors %}
        <p>Your username and password didn't match. Please try again.</p>
    {% endif %}

    <form method="post" action="{% url 'login' %}">
    {% csrf_token %}
        <table>
        <tr>
            <td>{{ form.username.label_tag }}</td>
            <td>{{ form.username }}</td>
        </tr>
        <tr>
```

```
            <td>{{ form.password.label_tag }}</td>
            <td>{{ form.password }}</td>
        </tr>
        </table>

        <input type="submit" value="login" />
        <input type="hidden" name="next" value="{{ next }}" />
    </form>
{% endblock %}
```

Verás que también hace uso de la plantilla base para mantener el estilo de tu blog.

La cosa buena aquí es que funciona. No necesitamos lidiar con el manejo de formularios, las contraseñas y asegurarlas. Solamente una cosa mas para hacer. Entonces vamos a la configuración en `mysite/settings.py`:

```
LOGIN_REDIRECT_URL = '/'
```

Entonces cuando se accede a la página directamente, la redireccionará a la página de primer nivel, el index (la página de inicio de blog).

Mejorando el diseño

Ya definimos como autorizar usuarios. Vemos los botones para añadir post. Ahora queremos asegurarnos que el botón de ingreso le aparezca a todos.

Vamos a agregar el botón de ingreso así:

```
<a href="{% url 'login' %}" class="top-menu"><span class="glyphicon glyphicon-lock"></span></a>
```

Para esto necesitamos editar la plantilla, así que vamos a abrir `blog/templates/blog/base.html` y cambiarlo, así que la parte en medio de `<body>` se verá así:

```
<body>
    <div class="page-header">
        {% if user.is_authenticated %}
            <a href="{% url 'post_new' %}" class="top-menu"><span class="glyphicon glyphicon-plus"></span></a>
            <a href="{% url 'post_draft_list' %}" class="top-menu"><span class="glyphicon glyphicon-edit"></span></a>
        {% else %}
            <a href="{% url 'login' %}" class="top-menu"><span class="glyphicon glyphicon-lock"></span></a>
        {% endif %}
        <h1><a href="/">Django DAW Blog</a></h1>
    </div>
    <div class="content container">
        <div class="row">
            <div class="col-md-8">
            {% block content %}
            {% endblock %}
            </div>
        </div>
    </div>
</body>
```

Debes reconocer el patrón aquí. Hay una condición *if* en la plantilla que verifica si el usuario está autenticado para mostrar los botones de agregar y editar. De otra manera muestra el botón de ingreso.

Más para usuarios autenticados

Vamos a agregarle un poco de azúcar a nuestras plantillas mientras estamos en esto. Primero vamos a mostrar algunos detalles de cuando ingresamos. Editemos el archivo `blog/templates/blog/base.html` así:

```
<div class="page-header">
    {% if user.is_authenticated %}
        <a href="{% url 'post_new' %}" class="top-menu"><span class="glyphicon glyphicon-plus"></span></a>
        <a href="{% url 'post_draft_list' %}" class="top-menu"><span class="glyphicon glyphicon-edit"></span></a>
        <p class="top-menu">Hello {{ user.username }} <small>(<a href="{% url 'logout' %}">Log out</a>)</small></p>
    {% else %}
        <a href="{% url 'login' %}" class="top-menu"><span class="glyphicon glyphicon-lock"></span></a>
    {% endif %}
    <h1><a href="/">Django DAW Blog</a></h1>
</div>
```

Esto agrega un mensaje "Hello " para recordarle al usuario como ingresó, y que está autenticado. También agrega un enlace de salida del blog – como puedes ver, aún no funciona.

Decidimos apoyarnos en Django para manejar el ingreso, así que vamos a dejar que Django se encargue de la salida. Mira https://docs.djangoproject.com/en/1.11/topics/auth/default/ y ve si encuentras algo.

Ya que termines de leer. Vamos a pensar en agregar una URL en `mysite/urls.py` apuntando a la vista de salida (`django.contrib.auth.views.logout`) así:

```
from django.conf.urls import include, url
from django.contrib import admin

from django.contrib.auth import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^accounts/login/$', views.login, name='login'),
    url(r'^accounts/logout/$', views.logout, name='logout', kwargs={'next_page': '/'}),
    url(r'', include('blog.urls')),
]
```

Si has seguido todo lo anterior, en este punto, tu blog ahora:

- necesita un nombre de usuario y contraseña para ingresar,
- necesita haber ingresado para agregar, editar, publicar o eliminar posts
- puede salir de nuevo