Usuarios

Antonio Espín Herranz

Habilitar autenticación

- Al igual que ocurre con otras utilidades de Django la autenticación viene como una aplicación aparte y tenemos que comprobar si está instalada o no.

- Por defecto, el asistente la instala, pero se puede comprobar en **settings.py**. En **aplicaciones instaladas** y en el **middleware**.

Habilitar autenticación

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'model_user',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

django.contrib.auth

- Este paquete de Django aporta componentes para el sistema de autenticación de Django.
  - https://docs.djangoproject.com/es/4.2/ref/contrib/auth/

- El **modelo de usuario** dispone de las siguientes **Clases**:
  - **models.User**: Un usuario identificado.
  - **models.AnonymousUser**: Usuario NO identificado
  - **models.UserManager**: Métodos para crear users.
  - **models.Permission**: Representa un permiso
  - **models.Group**: Un grupo de usuarios
  - **Relaciones many to many entre**:
    - Usuarios y grupos
    - Usuarios y permisos

Métodos de User

Método	Descripción
is_authenticated()	Siempre devuelve True para usuario reales. Es una forma de determinar si el usuario se ha identificado. esto no implica que posea ningún permiso, y tampoco comprueba que la cuenta esté activa. Sólo indica que el usuario se ha identificado con éxito.
is_anonymous()	Devuelve True sólo para usuarios anónimos, y False para usuarios "reales". En general, es preferible usar el método is_authenticated().
get_full_name()	Devuelve la concatenación de los campos first_name y last_name, con un espacio en medio.
set_password(passwd)	Cambia la contraseña del usuario a la cadena de texto en claro indicada, realizando internamente las operaciones necesarias para calcular el código de comprobación o *hash* necesario. Este método *no* guarda el objeto User.
check_password(passwd)	devuelve True si la cadena de texto en claro que se le pasa coincide con la contraseña del usuario. Realiza internamente las operaciones necesarias

Métodos de User

	para calcular los códigos de comprobación o *hash* necesarios.
get_group_permissions()	Devuelve una lista con los permisos que tiene un usuario, obtenidos a través del grupo o grupos a las que pertenezca.
get_all_permissions()	Devuelve una lista con los permisos que tiene concedidos un usuario, ya sea a través de los grupos a los que pertenece o bien asignados directamente.
has_perm(perm)	Devuelve True si el usuario tiene el permiso indicado. El valor de perm está en el formato `"package.codename". Si el usuario no está activo, siempre devolverá False.
has_perms(perm_list)	Devuelve True si el usuario tiene *todos* los permisos indicados. Si el usuario no está activo, siempre devolverá False.

Métodos de User

has_module_perms(app_label)	Devuelve True si el usuario tiene algún permiso en la etiqueta de aplicación indicada, app_label. Si el usuario no está activo, siempre devolverá False.
get_and_delete_messages()	Devuelve una lista de mensajes (objetos de la clase Message) de la cola del usuario, y los borra posteriormente.
email_user(subj, msg)	Envía un correo electrónico al usuario. El mensaje aparece como enviado desde la dirección indicada en el valor DEFAULT_FROM_EMAIL. Se le puede pasar un tercer parámetro opcional, from_email, para indicar otra dirección de remite distinta.

Funcionamiento Grupos / Permisos

- # Asignar un usuario a un grupo:
  - >>>myuser**.groups = group_list**
- # Agregar un usuario a un grupo:
  - >>>myuser**.groups.add(group1, group2,...)**
- # Quitar un usuario de algún grupo:
  - >>>myuser**.groups.remove(group1, group2,...)**
- #Quitar un usuario de todos los grupos:
  - >>>myuser**.groups.clear()**
- #Los permisos trabajan de la misma forma
  - >>>myuser**.permissions = [permission_list]**
  - >>>myuser**.permissions.add(permission1, permission2, ...)**
  - >>>myuser**.permissions.remove(permission1, permission2, ...)**
  - >>>myuser**.permissions.clear()**

Iniciar sesión a mano

- Django proporciona dos funciones para crear la session, en el paquete: **django.contrib.auth**
  - **authenticated**():
    - identificador usuario y password, si ok → User, en caso contrario None.

```
In [1]: from django.contrib import auth

In [2]: user = auth.authenticate(username='antonio',password='antonio')

In [3]: user
Out[3]: <User: antonio>

In [4]: user.is_staff
Out[4]: True
```

  - **login**(): La anterior sólo comprueba las credenciales, ahora hay que hacer una llamada a login() **para abrir la sesión**.

Abrir sesión en una Vista

```
from django.contrib import auth

def vista_login(request):
    username = request.POST.get('username', '')
    password = request.POST.get('password', '')
    user = auth.authenticate(username=username, password=password)
    if user is not None and user.is_active:
        # Contraseña correcta y usuario marcado como "activo"
        auth.login(request, user)
        # Redireccciona a una página de entrada correcta.
        return HttpResponseRedirect("/account/loggedin/")
    else:
        # Muestra una página de error
        return HttpResponseRedirect("/account/invalid/")
```

Cerrar sesión en una Vista

```
from django.contrib import auth

def logout(request):
    auth.logout(request)
    # Redirecccciona a una página de entrada correcta.
    return HttpResponseRedirect("/account/loggedout/")
```

- Si se llama logout y no hay ningún usuario conectado no ocurre nada.

Vistas / Ubicación plantillas

- El sistema de autentificación viene con un par de vistas para realizar las tareas de autentificación.

  - **LoginView** / **LogoutOut**
  - **En:** django.contrib.auth.views

  - Se pueden añadir nuestras **plantillas**.
    - En este caso las plantillas tienen que estar en:
    - Miapp
      - templates
        - » registration
          - **login.html**
          - **logout.html**

URL Login

- Para el **login**:
  - **re_path**('^accounts/login/$', LoginView.as_view(), name='login'),

- La página destino si va bien el login:
    - re_path('^accounts/profile/$', model_user.views.getInfoUser, name='profile'),

  - Coincide con la petición inicial:
    - path('', model_user.views.getInfoUser, name='getinfo'),

- La plantilla de **login.html**:

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

    {% block content %}
      <h2>Login (manual)</h2>
      <form method="post">
      {% csrf_token %}
      {{ form.as_p }}
      <button type="submit">Login</button>
      </form>
{% endblock %}
```

URL logout

- re_path('^accounts/logout/$', LogoutView.as_view(template_name='registration/logout.html',**next_page='/accounts/profile**'), name='logout'),

- *Con **next_page**, se indica la página la siguiente página cuando hacemos logout, pero en este caso poner la plantilla es redundante porque no se para en la plantilla, va directo a esa página.*

- La plantilla logout.html

```
{% extends "model_user/base.html" %}
    {% block content %}
    <p>Logged out!</p>
    <a href="{% url 'login' %}">Login de nuevo</a> | <a href="{% url 'getinfo' %}">Home</a>
{% endblock %}
```

Limitar el acceso a los usuarios

- Capturando el usuario de la request podemos saber si este está autenticado o no.

- Si el usuario NO está autenticado se redirige a la página de login.
  - Con el parámetro next indicamos a donde tiene que ir cuando se identifique:

```
from django.shortcuts import redirect

def mi_vista(request):
    if not request.user.is_authenticated():
        return redirect('/login/?next=%s' % request.path)
    # ...
```

Decoradores

- from django.contrib.auth.decorators import login_required

- @login_required
- def mi_vista(request):
  - # ...

- El decorador controla si el usuario está validado, si está validado ejecuta la vista tal cual.

- Si no está validado le manda a la pantalla de login para que se identifique.

Decoradores

- Comprobar si el usuario está autenticado y además tiene cierto permiso.
- Django proporciona el decorador @user_passes_test(un_permiso, login_url="")

- Ejemplo (**forma manual**)

```
def puede_votar(user):
    return user.is_authenticated() and user.has_perm("biblioteca.puede_votar")

@user_passes_test(puede_votar, login_url="/login/")
def votar(request):
    …
```

Decoradores

- from django.contrib.auth.decorators import permission_required

- @permission_required('biblioteca.puede_votar', login_url="/login/")
- def votar(request):
  - # ...

- El decorador **permission_required**() también acepta el parámetro opcional login_url, de nuevo con el valor **/accounts/login/ en caso de omisión**.

Proteger vistas genéricas

- La forma más simple es poner un decorador en la **URLConf**, en el **método as_view()**

```
from django.contrib.auth.decorators import login_required, permission_required
from django.views.generic import TemplateView

path('pagina1/', login_required(TemplateView.as_view(template_name='model_user/pagina1.html')),
    name='pagina1'),

path('escuchar_cancion/',
    permission_required('model_user.escuchar_cancion')(TemplateView.as_view(template_name='model_user
    /escuchar_cancion.html')), name='escuchar_cancion'),

path('bajar_cancion/',
    permission_required('model_user.bajar_cancion')(TemplateView.as_view(template_name='model_user/ba
    jar_cancion.html')), name='bajar_cancion'),
```

- **model_user**: es la app.
- ***Los permisos se pueden crear en el model:***
- **Después se otorgan con el panel de Administración**

Lanzar el comando:
makemigrations / migrate

```
class Cancion(models.Model):
    titulo=models.CharField(max_length=25)
    grupo=models.CharField(max_length=30)

    class Meta:
        permissions=(('escuchar_cancion','escuchar cancion'), \
        ('bajar_cancion','bajar cancion'))
```

Proteger vistas genéricas

- Si la vista genérica hay que protegerla en la clase, tenemos que colocar el decorador en el método dispatch de la clase, hay que sobrescribirlo:

```
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator
from django.views.generic import TemplateView

class VistaProtegida(TemplateView):
    template_name = 'secreto.html‘

    @method_decorator(login_required)
    def dispatch(self, *args, **kwargs):
        return super(VistaProtegida, self).dispatch(*args, **kwargs)
```

Model User

Otras clases

django.contrib.auth

- **models.UserManager**: añade métodos para crear usuarios:
  - **create_user**(username, email=None, password=None, **extra_fields)
  - **create_superuser**(username, email, password, **extra_fields)
    - Establece is_staff / is_superuser a True

django.contrib.auth

- **models.AnonymousUser**
  - Representa un usuario no autenticado. Si capturamos de la request el usuario.
  - **request.user / get_user(request)** → Devolverá un **AnonymousUser**
    - id is always None.
    - username: cadena vacía
    - get_username() cadena vacía
    - is_anonymous: True .
    - is_authenticated: False
    - is_staff and is_superuser: False
    - is_active: False
    - groups and user_permissions:vacíos
    - set_password(), check_password(), save() and delete() raise NotImplementedError.

django.contrib.auth

- **model.Permission**: Permisos
  - **name**:
    - Requirido, 255 chars. 'Can vote'.
  - **content_type**:
    - Requerido. Referencia django_content_type (Tabla de la BD) que contiene un registro de cada modelo instalado.
  - **codename**:
    - Requerido 100 char, ejemplo 'can_vote'.
  - Soporta los métodos de objects

django.contrib.auth

- **model.Group**:
  - Grupos de usuarios.
    - name: Nombre del grupo. 80 chars. Requerido
    - permissions: Many To many con Permission

    - group.permissions.set([permission_list])
    - group.permissions.add(permission, permission, ...)
    - group.permissions.remove(permission, permission, ...)
    - group.permissions.clear()

Enlaces

- https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Authentication