Django
Python Social Auth

By German Quiroz Bogner

Python-Social-Auth es una aplicación para Django que permite autenticar los usuarios a través de las redes sociales tales como Google, Twitter, Facebook, entre otros.

Para usar esta aplicación, se debe introducir la siguiente instrucción en la línea de comandos para instalarla:

```
pip install python-social-auth
```

La registramos en nuestro proyecto en la sección de `INSTALLED_APPS` del archivo "*settings.py*".

```
INSTALLED_APPS = (
	...
	'social.apps.django_app.default',
	...
)
```

Necesitamos registrarla en `TEMPLATE_CONTEXT_PROCESSORS` para poder utilizarla en nuestros templates.

```
TEMPLATE_CONTEXT_PROCESSORS = (
	...
	'social.apps.django_app.context_processors.backends',
	'social.apps.django_app.context_processors.login_redirect',
	...
)
```

Utilizar python-Social-Auth con la autenticación de Google en pocos pasos. En primer lugar debemos indicarle a nuestro proyecto que backends va a utilizar para la autenticación. En nuestro archivo *"settings.py"* agregamos lo siguiente:

```
AUTHENTICATION_BACKENDS = (
	...
	'django.contrib.auth.backends.ModelBackend',
	'social.backends.google.GoogleOAuth2',
	...
)
```

Una vez configurado esto debemos ir a la Google Developer Console e iniciar un proyecto nuevo, indicando que es para una aplicación web.

Con el proyecto iniciado nos dirigimos a la sección de API's de la consola y habilitamos a la API de Google+. En la sección de credenciales creamos una para nuestro proyecto indicando cual es el dominio que utilizamos y la url donde se redirecciona al usuario si el login es exitoso.

Por ultimo indicamos estos datos en nuestro archivo "*settings.py*":

```
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = 'id-de-cliente'
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = 'codigo-secreto'
```

¡Ya tenemos el sistema de autenticación configurado! Veamos unas configuraciones extras:

- Utilizando la variable `SOCIAL_AUTH_LOGIN_REDIRECT_URL` podemos definir la url a la que el usuario será redirigido si el login fue exitoso.

```
SOCIAL_AUTH_LOGIN_REDIRECT_URL = '/url-donde-se-redirige/'
```

- Podemos limitar los dominios de eMail e indicar sólo los que nos interese usar, seteando la variable `WHITELISTED_DOMAINS`.

```
WHITELISTED_DOMAINS = ['gmail.com', 'jacana-soft.com']
```

- Utilizando pipelines se puede gestionar que es lo que sucede con la información del usuario una vez que el login fue exitoso. De manera predeterminada, Python-Social-Auth tiene el siguiente comportamiento:

```
SOCIAL_AUTH_PIPELINE = (
      # recibe vía backend y uid las instancias de social_user y user
      'social.pipeline.social_auth.social_details',
      'social.pipeline.social_auth.social_uid',
      'social.pipeline.social_auth.auth_allowed',

      # Recibe según user.email la instancia del usuario y lo
      reemplaza con uno que recibió anteriormente
      'social.pipeline.social_auth.social_user',

      # Trata de crear un username válido según los datos que recibe
      'social.pipeline.user.get_username',

      # Crea un usuario nuevo si uno todavía no existe
      'social.pipeline.user.create_user',

      # Trata de conectar las cuentas
      'social.pipeline.social_auth.associate_user',
```

```
        # Recibe y actualiza social_user.extra_data
        'social.pipeline.social_auth.load_extra_data',

        # Actualiza los campos de la instancia user con la información
        que obtiene vía backend
        'social.pipeline.user.user_details',
)
```

El paso siguiente es agregar las url de python-social=auth a nuestra lista de url del proyecto en el archivo "*urls.py*":

```
urlpatterns = patterns (
        '',
        url(
                r'^social/', include('social.apps.django_app.urls',
        namespace='social')
        ),
)
```

Por último sincronizamos la base de datos con el comando syncdb:

```
        >>>python manage.py syncdb
```

Para mostrar el botón de autenticación en nuestra planilla usamos:

```
<a href="{% url 'social:begin' 'google-oauth2' %}">
        Entrar con la cuenta de Google
</a>
```