Chapter 8: Custom User Model

يسمح لنا نموذج المستخدم المدمج في Django ببدء العمل مع المستخدمين على الفور ، كما فعلنا للتو مع تطبيق المدونة الخاص بنا في الفصول السابقة. ومع ذلك ، توصي وثائق Django الرسمية بشدة باستخدام نموذج مستخدم مخصص للمشاريع الجديدة. والسبب هو أنه إذا كنت ترغب في إجراء أي تغييرات على نموذج المستخدم على الطريق – على سبيل المثال إضافة حقل العمر – فإن استخدام نموذج مستخدم مخصص من البداية يجعل هذا الأمر سهلا للغاية. ولكن إذا لم تقم بإنشاء نموذج مستخدم مخصص ، فإن تحديث نموذج المستخدم الافتراضي في مشروع Django موجود يمثل تحديا كبيرا جدا. لذلك استخدم دائما نموذج مستخدم مخصص لجميع مشاريع Django الجديدة. لكن النهج الموضح في مثال الوثائق الرسمية ليس في الواقع ما يوصي به العديد من خبراء جانغو. إنه يستخدم AbstractBaseUser المعقد للغاية عندما نستخدم AbstractUser بدلا من ذلك ، تكون الأمور أبسط بكثير ولا تزال قابلة للتخصيص. وبالتالي سنستخدم AbstractUser في هذا الفصل حيث نبدأ تطبيق صحيفة جديد بشكل صحيح مع متغيرات البيئة و...

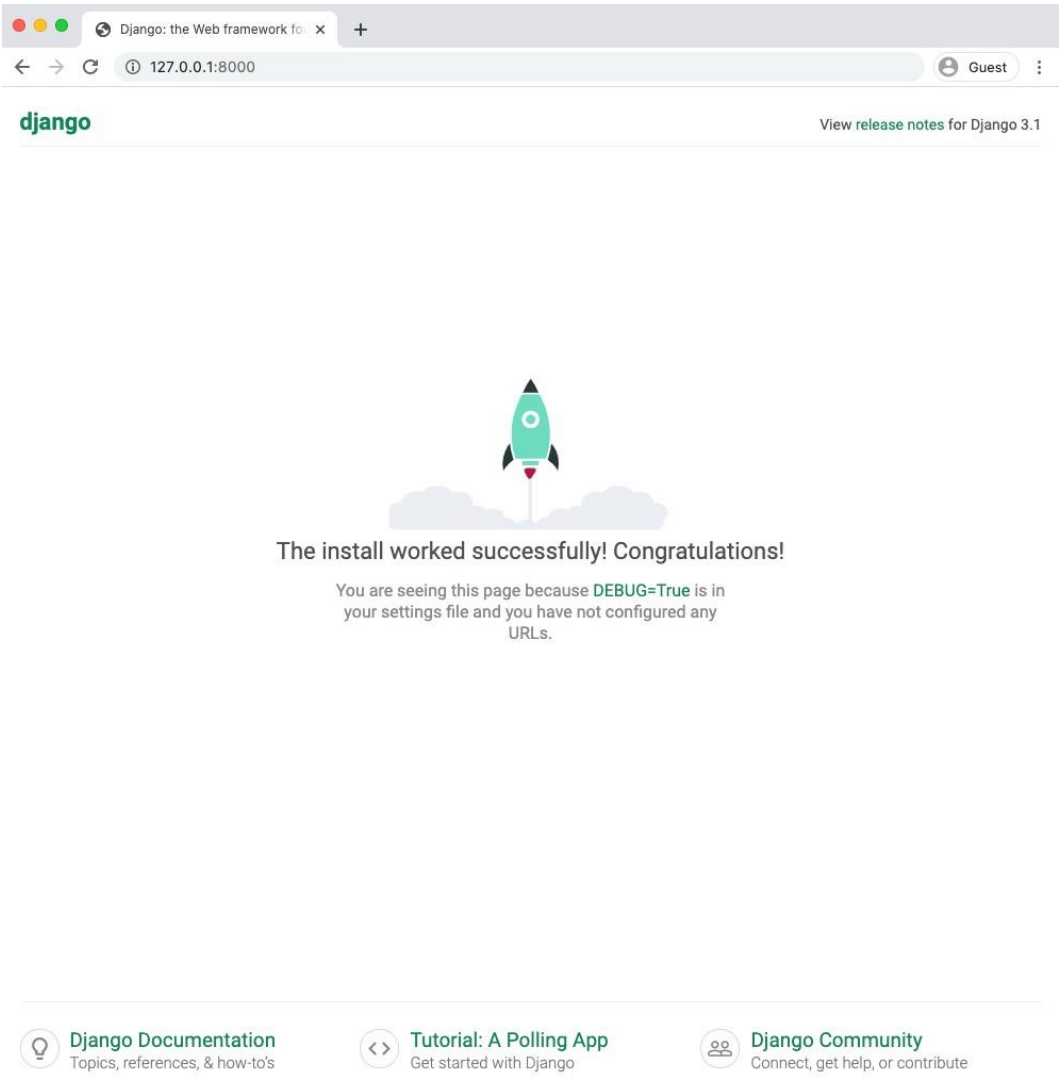
Here are the commands to run:

Command Line

```
$ cd ~/Desktop
$ mkdir news
$ cd news
$ pipenv install django~=3.1.0
$ pipenv shell
(news) $ django-admin startproject config .
(news) $ python manage.py startapp accounts
(news) $ python manage.py runserver
```

لاحظ أننا لم نقم بتشغيل الترحيل لتكوين قاعدة البيانات الخاصة بنا. من المهم الانتظار حتى بعد إنشاء نموذج المستخدم المخصص الجديد قبل القيام بذلك نظرا لمدى ارتباط نموذج المستخدم بإحكام ببقية.Django

If you navigate to `http://127.0.0.1:8000` you'll see the familiar Django welcome screen.



Welcome page

Custom User Model

يتطلب إنشاء نموذج المستخدم المخصص أربع خطوات:

- تحديث التكوين / **setting.py**
- إنشاء نموذج **CustomUser** جديد إنشاء نماذج جديدة ل **UserCreationForm** و **UserChangeForm**
- تحديث الحسابات / **admin.py** في التكوين / **setting.py** سنضيف تطبيق الحسابات إلى **INSTALLED_APPS** لدينا. ثم في الجزء السفلي من الملف ، استخدم تكوين **AUTH_USER_MODEL** لإخبار **Django** باستخدام نموذج المستخدم المخصص الجديد بدلا من نموذج المستخدم المدمج. سنسمي نموذج المستخدم المخصص الخاص بنا **CustomUser** لذلك ، نظرا لوجوده داخل تطبيق الحسابات الخاص بنا ، فإننا نشير إليه باسم 'accounts'. مستخدم مخصص

.Code

```
# config/settings.py INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'accounts', # new
] ...
AUTH_USER_MODEL = 'accounts.CustomUser' # new
```

الآن قم بتحديث accounts/models.py باستخدام نموذج مستخدم جديد سنسميه CustomUser الذي يوسع AbstractUser الحالي.

نقوم أيضا بتضمين أول حقل مخصص لدينا ، age ، هنا.

Code

```
# accounts/models.py from django.contrib.auth.models import AbstractUser
from django.db import models


class CustomUser(AbstractUser):
    age = models.PositiveIntegerField(null=True, blank=True)
```

إذا قرأت الوثائق الرسمية حول نماذج المستخدم المخصصة ، فإنها توصي باستخدام AbstractBaseUser وليس AbstractUser. هذا يعقد الأمور بلا داع في رأيي، خاصة بالنسبة للمبتدئين.

AbstractBaseUser vs AbstractUser

يتطلب AbstractBaseUser مستوى جيدا جدا من التحكم والتخصيص. نحن أساسا إعادة كتابة جانغو. يمكن أن يكون هذا مفيدا ، ولكن إذا كنا نريد فقط نموذج مستخدم مخصص يمكن تحديثه بحقول إضافية ، فإن الخيار الأفضل هو AbstractUser الذي يصنف AbstractBaseUser.بمعنى آخر ، نكتب تعليمات برمجية أقل بكثير ولدينا فرصة أقل لإفساد الأمور. إنه الخيار الأفضل ما لم تكن تعرف حقا ما تفعله مع Django!

لاحظ أننا نستخدم كلا من **null**122 و **blank** مع حقل العمر الخاص بنا. من السهل الخلط بين هذين المصطلحين ولكنهما مختلفان تماما :

Null مرتبط بقاعدة البيانات. عندما يحتوي الحقل على **null = True** ، يمكنه تخزين إدخال قاعدة بيانات ك **NULL** ، مما يعني عدم وجود قيمة. فارغ مرتبط بالتحقق من الصحة ،

إذا كان **Blank = True**، فسيسمح النموذج بقيمة فارغة ، بينما إذا كان **null = False**، فستكون القيمة مطلوبة. في الممارسة العملية ، يتم استخدام **null** و **blank** معا بشكل شائع بهذه الطريقة بحيث يسمح النموذج بقيمة فارغة وتخزن قاعدة البيانات هذه القيمة على أنها **NULL**.من الأمور الشائعة التي يجب أن تكون على دراية بها هو أن نوع الحقل يملي كيفية استخدام هذه القيم. عندما يكون لديك حقل يستند إلى سلسلة مثل **CharField** أو **TextField** ، فإن تعيين كل من **blank** و **null** كما فعلنا سيؤدي إلى قيمتين محتملتين ل "لا توجد بيانات" في قاعدة البيانات. وهي فكرة سيئة. بدلا من ذلك ، فإن اتفاقية **Django** هي استخدام **empty string** **''** , وليس **NULL**.

إذا عدنا خطوة إلى الوراء للحظة ، فما هي الطريقتان اللتان سنتفاعل بهما مع نموذج CustomUser الجديد الخاص بنا؟ إحدى الحالات هي عندما يقوم المستخدم بالتسجيل للحصول على حساب جديد على موقعنا. والآخر داخل تطبيق المشرف الذي يسمح لنا ، كمستخدمين متميزين ، بتعديل المستخدمين الحاليين. لذلك سنحتاج إلى تحديث النموذجين المدمجين لهذه الوظيفة :UserCreationForm وUserChangeForm

Stop the local server with `Control+c` and create a new file in the `accounts` app called `forms.py`.

Command Line

```
(news)$touchaccounts/forms.py
```

We'll update it with the following code to extend the existing `UserCreationForm` and `UserChangeForm` forms.

Code

```
# accounts/forms.py from django import forms from django.contrib.auth.forms import
UserCreationForm, UserChangeForm from .models import CustomUser


class CustomUserCreationForm(UserCreationForm):

    class Meta(UserCreationForm):
        model = CustomUser
        fields = UserCreationForm.Meta.fields + ('age',)
class CustomUserChangeForm(UserChangeForm):

    class Meta:
        model = CustomUser
        fields = UserChangeForm.Meta.fields
```

بالنسبة لكلا النموذجين الجديدين ، نستخدم Meta class126 لتجاوز الحقول الافتراضية عن طريق تعيين النموذج إلى CustomUser واستخدام الحقول الافتراضية عبر Meta.fields التي تتضمن جميع الحقول الافتراضية. لإضافة حقل العمر المخصص الخاص بنا ، نقوم ببساطة بمعالجته في النهاية وسيتم عرضه تلقائيا على صفحة التسجيل المستقبلية. بقعة جميلة ، لا؟ يمكن أن يكون مفهوم الحقول في النموذج مربكا في البداية ، لذلك دعونا نأخذ لحظة لاستكشافه أكثر. يحتوي نموذج CustomUser الخاص بنا على جميع حقول نموذج المستخدم

الافتراضي وحقل العمر الإضافي الذي قمنا بتعيينه. ولكن ما هي هذه الحقول الافتراضية؟ اتضح أن هناك العديد من ١٢٧ بما في ذلك اسم المستخدم first_name last_name والبريد الإلكتروني وكلمة المرور والمجموعات والمزيد. ومع ذلك ، عندما يقوم المستخدم بالتسجيل للحصول على حساب جديد على Django ، فإن النموذج الافتراضي يطلب فقط اسم مستخدم وبريد إلكتروني وكلمة مرور. يخبرنا هذا أن الإعداد الافتراضي للحقول على UserCreationForm هو مجرد اسم مستخدم وبريد إلكتروني وكلمة مرور على الرغم من وجود العديد من الحقول المتاحة. ...الخطوة الأخرى الوحيدة التي نحتاجها هي تحديث ملف admin.py الخاص بنا نظرا لأن المسؤول مقترن بإحكام بنموذج المستخدم الافتراضي. سنقوم بتوسيع فئة UserAdmin[128] الحالية لاستخدام نموذج CustomUser الجديد الخاص بنا.

[126] https://docs.djangoproject.com/en/3.1/topics/forms/modelforms/#overriding-the-default-fields

[127] https://docs.djangoproject.com/en/3.1/ref/contrib/auth/#django.contrib.auth.models.User

[128] https://docs.djangoproject.com/en/3.1/topics/auth/customizing/#extending-the-existing-user-model

Code

```
# accounts/admin.py from django.contrib import admin from django.contrib.auth.admin import
UserAdmin from .forms import CustomUserCreationForm, CustomUserChangeForm from .models
import CustomUser


class CustomUserAdmin(UserAdmin): add_form =
    CustomUserCreationForm form = CustomUserChangeForm model =
    CustomUser


admin.site.register(CustomUser, CustomUserAdmin)
```

حسنا لقد انتهينا! اكتب **Control+c** لإيقاف الخادم المحلي والمضي قدما وتشغيل **makemigrations** والترحيل لأول مرة لإنشاء قاعدة بيانات جديدة تستخدم نموذج المستخدم المخصص.

Command Line

```
(news) $ python manage.py makemigrations accounts Migrations for 'accounts':
  accounts/migrations/0001_initial.py
    - Create model CustomUser (news) $ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, users Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK Applying
  auth.0008_alter_user_username_max_length... OK
```

```
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying accounts.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying sessions.0001_initial... OK
```

Superuser

دعنا ننشئ حساب **superuser** للتأكد من أن كل شيء يعمل كما هو متوقع. في سطر الأوامر اكتب الأمر التالي وانتقل عبر سطر الأوامر.

Command Line

```
(news)$pythonmanage.pycreatesuperuser
```

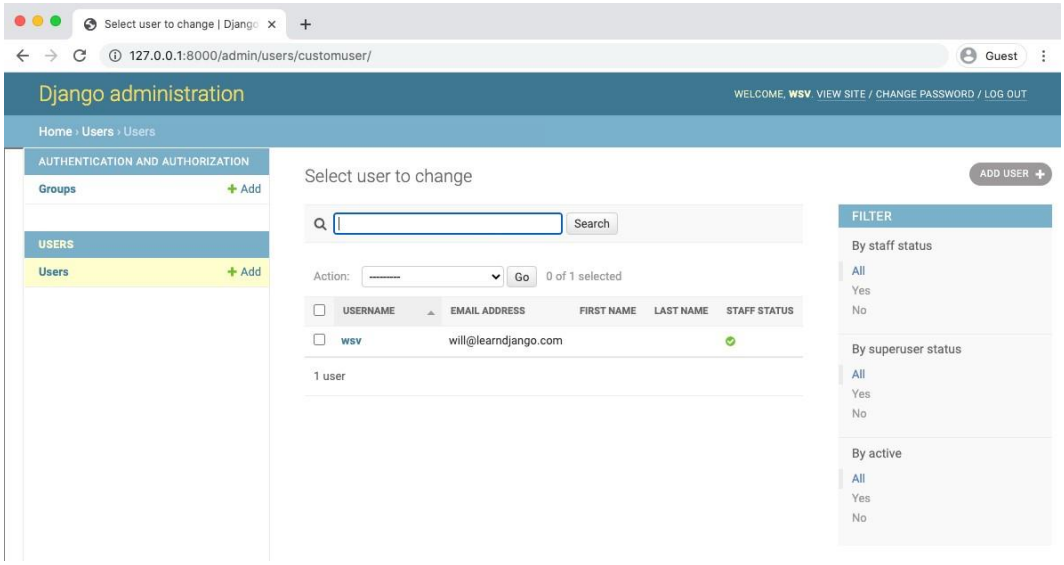
حقيقة أن هذا يعمل هو أول دليل على أن نموذج المستخدم المخصص لدينا يعمل كما هو متوقع. دعونا نعرض الأشياء في المتصفح أيضا للتأكد من ذلك. بدء تشغيل خادم الويب.

Command Line

```
(news)$pythonmanage.pyrunserver
```

Then navigate to the admin at `http://127.0.0.1:8000/admin` and log in.

إذا قمت بالنقر فوق الرابط الخاص ب "المستخدمون" ، فيجب أن ترى حساب superuser الخاص بك بالإضافة إلى الحقول الافتراضية لاسم المستخدم وعنوان البريد الإلكتروني والاسم الأول واسم العائلة وحالة الموظفين.



Admin one user

للتحكم في الحقول المدرجة هنا ، نستخدم **list_display**. ومع ذلك ، لتحرير حقول مخصصة جديدة وإضافتها بالفعل ، مثل العمر ، يجب علينا أيضا إضافة **fieldsets**[130].

Code

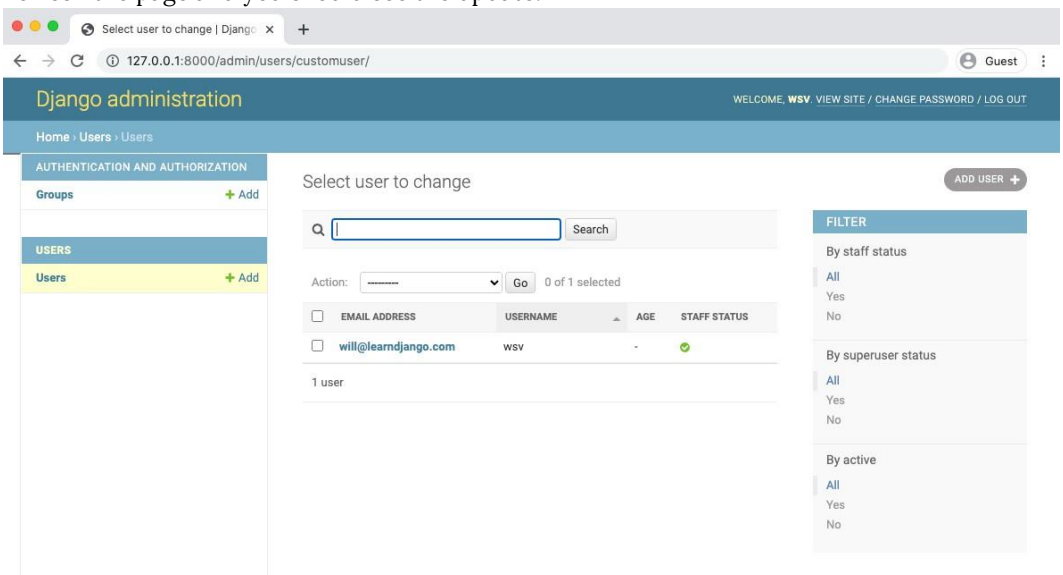
```
# accounts/admin.py from django.contrib import admin from
django.contrib.auth.admin import UserAdmin

from .forms import CustomUserCreationForm, CustomUserChangeForm from .models import CustomUser


class CustomUserAdmin(UserAdmin): add_form = CustomUserCreationForm form = CustomUserChangeForm
    model = CustomUser list_display = ['email', 'username', 'age', 'is_staff', ] # new
    fieldsets = UserAdmin.fieldsets + ( # new (None, {'fields': ('age',)}),
    )
```

[129]https://docs.djangoproject.com/en/3.1/ref/contrib/admin/#django.contrib.admin.ModelAdmin.list_display

[130]https://docs.djangoproject.com/en/3.1/topics/auth/customizing/#custom-users-and-django-contribadmin

```
    add_fieldsets = UserAdmin.add_fieldsets + ( # new (None, {'fields': ('age',)}),
    )


admin.site.register(CustomUser, CustomUserAdmin)
```

Refresh the page and you should see the update.



Admin custom list display

مع اكتمال نموذج المستخدم المخصص لدينا ، يمكننا الآن التركيز على بناء بقية موقع صحيفتنا. في الفصل التالي ، سنقوم بتكوين صفحات التسجيل وتسجيل الدخول وتسجيل الخروج وتخصيصها.

Chapter 9: User Authentication

الآن بعد أن أصبح لدينا نموذج مستخدم مخصص يعمل ، يمكننا إضافة الوظائف التي يحتاجها كل موقع ويب: القدرة على التسجيل وتسجيل الدخول وتسجيل خروج المستخدمين. يوفر **Django** كل ما نحتاجه لتسجيل الدخول وتسجيل الخروج ولكننا سنحتاج إلى إنشاء نموذج خاص بنا لتسجيل مستخدمين جدد. سنقوم أيضا بإنشاء صفحة رئيسية أساسية تحتوي على روابط إلى جميع الميزات الثلاث حتى لا نضطر إلى كتابة عناوين **URL** يدويا في كل مرة.

Templates

بشكل افتراضي، يبحث محمل قالب Django عن قوالب في بنية متداخلة داخل كل تطبيق. ستكون هناك حاجة إلى بنية accounts/templates/accounts/home.html لقالب home.html داخل تطبيق accounts. لكن دليل القوالب الواحدة ضمن نهج التكوين أنظف ويتدرج بشكل أفضل ، وهذا ما سنستخدمه. دعنا ننشئ مجلد جديد للقوالب وداخله مجلد `registration` لأن هذا هو المكان الذي سيبحث فيه Django عن قالب تسجيل الدخول.

Command Line

```
(news) $ mkdir templates
(news) $ mkdir templates/registration
```

الآن نحن بحاجة إلى إخبار Django عن هذا الدليل الجديد عن طريق تحديث التكوين `'DIRS'` in `config/settings.py`. ل

This is a one-line change.

```
# config/settings.py TEMPLATES = [
    { ...
        'DIRS': [str(BASE_DIR.joinpath('templates'))], # new ...
    }
]
```

إذا كنت تفكر في ما يحدث عند تسجيل الدخول أو تسجيل الخروج من أحد المواقع، إعادة توجيهك على الفور إلى صفحة لاحقة. نحن بحاجة إلى إخبار Django بمكان إرسال المستخدمين في كل حالة. تقوم إعدادات LOGIN_REDIRECT_URL وLOGOUT_REDIRECT_URL بذلك. سنقوم بتكوين كليهما لإعادة التوجيه إلى صفحتنا الرئيسية التي ستحتوي على عنوان URL المسمى "الصفحة الرئيسية". تذكر أنه عندما ننشئ مسارات عناوين URL الخاصة بنا ، لدينا خيار إضافة اسم إلى كل منها. لذلك عندما نجعل عنوان URL للصفحة الرئيسية ، سنحرص على تسميته 'home'.

Add these two lines at the bottom of the `config/settings.py` file.

Code

```
# config/settings.py
LOGIN_REDIRECT_URL = 'home' # new
```

```
LOGOUT_REDIRECT_URL = 'home' # new
```

Now we can create four new templates:

Command Line

```
(news) $ touch templates/base.html
(news) $ touch templates/home.html
(news) $ touch templates/registration/login.html
(news) $ touch templates/registration/signup.html
```

إليك رمز HTML لكل ملف لاستخدامه. سيتم توريث القاعدة من قبل كل قالب آخر في مشروعنا. باستخدام كتلة مثل {% block content %} يمكننا لاحقا تجاوز المحتوى فقط في هذا المكان في قوالب أخرى

```
.<!-- templates/base.html -->

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>{% block title %}Newspaper App{% endblock title %}</title>
</head>
<body>
  <main>
    {% block content %}
    {% endblock content %}
  </main>
</body>
</html>
```

Code

```
<!-- templates/home.html -->
{% extends 'base.html' %}

{% block title %}Home{% endblock title %}

{% block content %}
{% if user.is_authenticated %} Hi {{ user.username }}!
  <p><a href="{% url 'logout' %}">Log Out</a></p>
{% else %}
  <p>You are not logged in</p>
  <a href="{% url 'login' %}">Log In</a> |
  <a href="{% url 'signup' %}">Sign Up</a>
{% endif %}
{% endblock content %}
```

```
<!-- templates/registration/login.html -->
{% extends 'base.html' %}

{% block title %}Log In{% endblock title %}

{% block content %}
<h2>Log In</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
```

```
  <button type="submit">Log In</button>
</form>
{% endblock content %}
```

Code

```
<!-- templates/registration/signup.html -->
{% extends 'base.html' %}

{% block title %}Sign Up{% endblock title %}

{% block content %}
<h2>Sign Up</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Sign Up</button>
</form>
{% endblock content %}
```

تم الآن تعيين جميع قوالبنا. لا يزال يتعين عليك الذهاب إلى عناوين URL و views ذات الصلة.

URLs

لنبدأ بمسارات عناوين .URL في ملف التكوين/عناوينURL.py ، نريد أن يظهر قالب html. الرئيسية كصفحة رئيسية، ولكننا لا نريد إنشاء تطبيق صفحات مخصص حتى الآن. يمكننا استخدام اختصار استيراد TemplateView وتعيين template_name مباشرة في نمط عنوان URL الخاص بنا.

بعد ذلك ، نريد "تضمين" كل من تطبيق الحسابات وتطبيق المصادقة المدمج. والسبب هو أن تطبيق المصادقة المدمج يوفر بالفعل طرق عرض وعناوين URLلتسجيل الدخول وتسجيل الخروج. ولكن للتسجيل ، سنحتاج إلى إنشاء طريقة العرض وعنوان URL الخاص بنا. لضمان اتساق مسارات عناوين URL الخاصة بنا ، نضعها في accounts / بحيث تكون عناوين URL النهائية هي : accounts/login accounts/logout , accounts/signup

```
# config/urls.py from django.contrib import admin from django.urls import path,
include # new from django.views.generic.base import TemplateView # new

urlpatterns = [ path('admin/', admin.site.urls), path('accounts/', include('accounts.urls')),
    # new path('accounts/', include('django.contrib.auth.urls')), # new path('',
    TemplateView.as_view(template_name='home.html'),
      name='home'), # new
]
```

Now create a `urls.py` file in the `accounts` app.

Command Line

```
(news)$touchaccounts/urls.py
```

Update `accounts/urls.py` with the following code:

Code

```
# accounts/urls.py from django.urls import path from
.views import SignUpView
```

```
urlpatterns = [ path('signup/', SignUpView.as_view(), name='signup'),
]
```

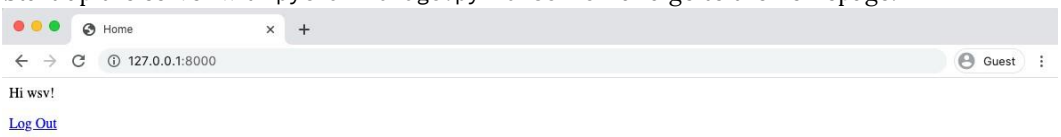
الخطوة الأخيرة هي ملف `views.py` الخاص بنا والذي سيحتوي على منطق `views.py` الخاص بنا. نحن نستخدم `CreateView` العام من `Django` هنا ونخبره باستخدام نموذج `CustomUserCreationForm` الخاص بنا ، لإعادة التوجيه إلى تسجيل الدخول بمجرد تسجيل المستخدم بنجاح ، وأن يتم تسمية القالب الخاص بنا.`signup.html`

Code

```
# accounts/views.py from django.urls import reverse_lazy from
django.views.generic import CreateView from .forms import
CustomUserCreationForm


class SignUpView(CreateView):
    form_class = CustomUserCreationForm success_url = reverse_lazy('login')
    template_name = 'registration/signup.html'
```

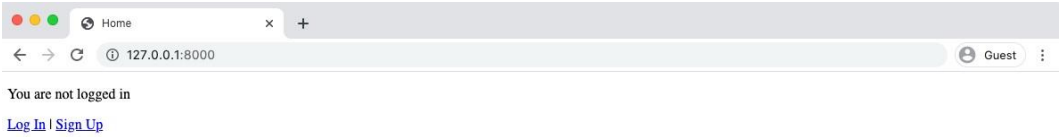
حسنا ، يا أخي! نحن فعلنا. دعونا نختبر الأمور.

Start up the server with `python manage.py runserver` and go to the homepage.



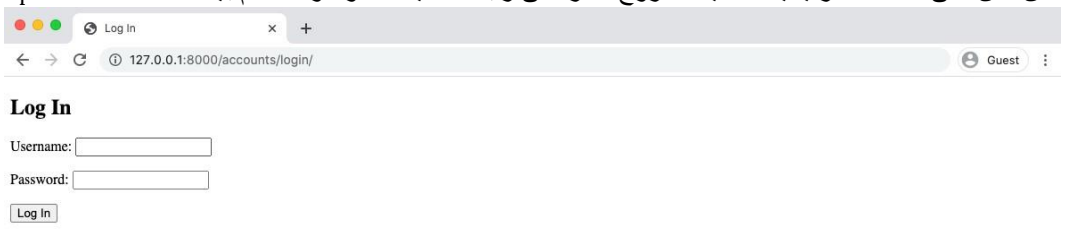
Homepage logged in

لقد قمنا بتسجيل الدخول إلى المسؤول في الفصل السابق ، لذا يجب أن ترى تحية مخصصة هنا.Click on the “Log Out” link.



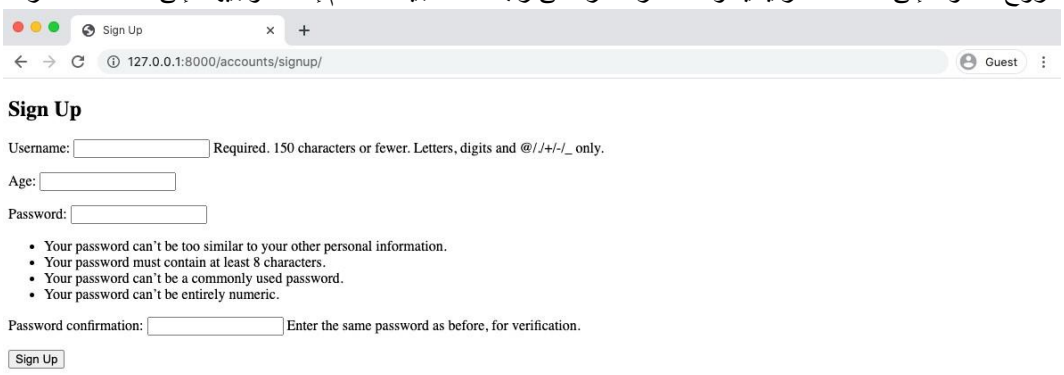
Homepage logged out

الآن نحن على الصفحة الرئيسية لتسجيل الخروج. انقر على رابط تسجيل الدخول واستخدام بيانات اعتماد superuser الخاص بك.



Log in

عند تسجيل الدخول بنجاح ، ستتم إعادة توجيهك مرة أخرى إلى الصفحة الرئيسية ومشاهدة نفس التحية المخصصة. إنه يجدي نفعًا! الآن استخدم رابط "تسجيل الخروج" للعودة إلى الصفحة الرئيسية وهذه المرة انقر على رابط "التسجيل". ستتم إعادة توجيهك إلى صفحة الاشتراك الخاصة بنا. تأكد من تضمين حقل العمر !



Sign up page

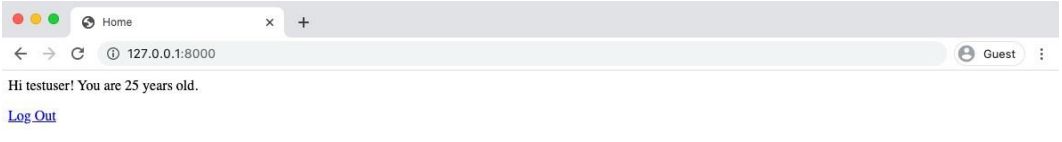
إنشاء مستخدم جديد. يسمى testuser ولقد حددت العمر إلى ٢٥. بعد إرسال النموذج بنجاح ، ستتم إعادة توجيهك إلى صفحة تسجيل الدخول. سجل الدخول باستخدام المستخدم الجديد وستتم إعادة توجيهك مرة أخرى إلى الصفحة الرئيسية مع تحية مخصصة للمستخدم الجديد. ولكن نظرا لأن لدينا حقل العمر الجديد ، دعنا نضيف ذلك إلى قالبhtml home. . إنه حقل على user model، لذلك لعرضه

we only need to use `{{ user.age }}`.

Code

```
<!-- templates/home.html -->
{% extends 'base.html' %}

{% block title %}Home{% endblock title %}

{% block content %}
{% if user.is_authenticated %}
  Hi {{ user.username }}! You are {{ user.age }} years old.
  <p><a href="{% url 'logout' %}">Log Out</a></p>
{% else %}
  <p>You are not logged in</p>
  <a href="{% url 'login' %}">Log In</a> |
  <a href="{% url 'signup' %}">Sign Up</a>
{% endif %}
{% endblock content %}
```

Save the file and refresh the homepage.

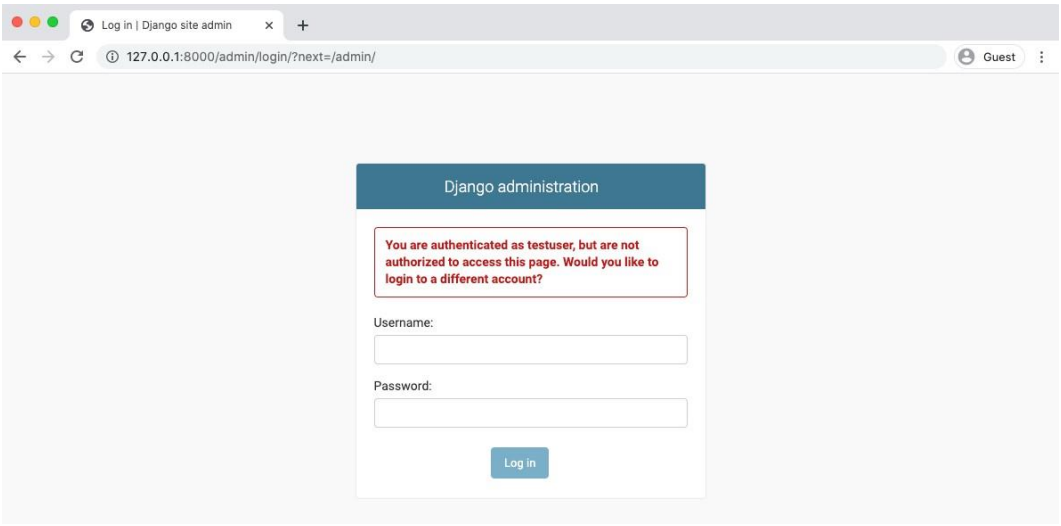


Homepage for testuser

Everything works as expected.

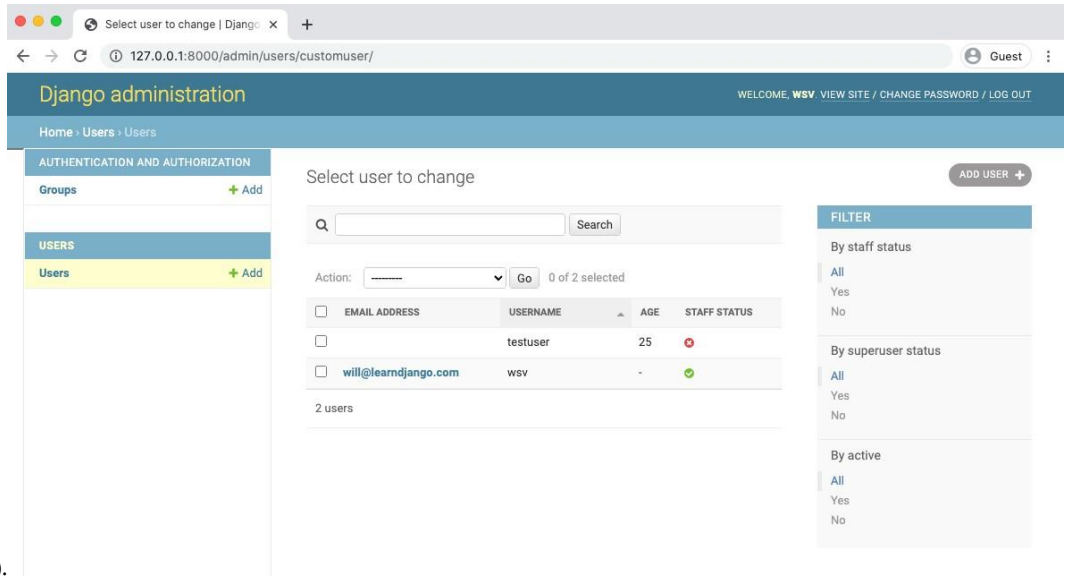
Admin

Let's also log in to the admin to view our two user accounts. Navigate to `http://127.0.0.1:8000/admin` and ...



Admin log in wrong

ما هذا! لماذا لا يمكننا تسجيل الدخول؟ حسنا ، لقد قمنا بتسجيل الدخول باستخدام حساب testuser الجديد وليس حساب superuser الخاص بنا. فقط حساب superuser لديه أذونات لتسجيل الدخول إلى Admin! لذا استخدم حساب superuser لتسجيل الدخول بدلا من ذلك. بعد الانتهاء من ذلك ، يجب أن ترى الصفحة الرئيسية العادية للمشرف. انقر فوق المستخدمين ويمكنك رؤية اثنين من المستخدمين: حساب testuser الذي أنشأناه للتو واسم superuser السابق (wsv).



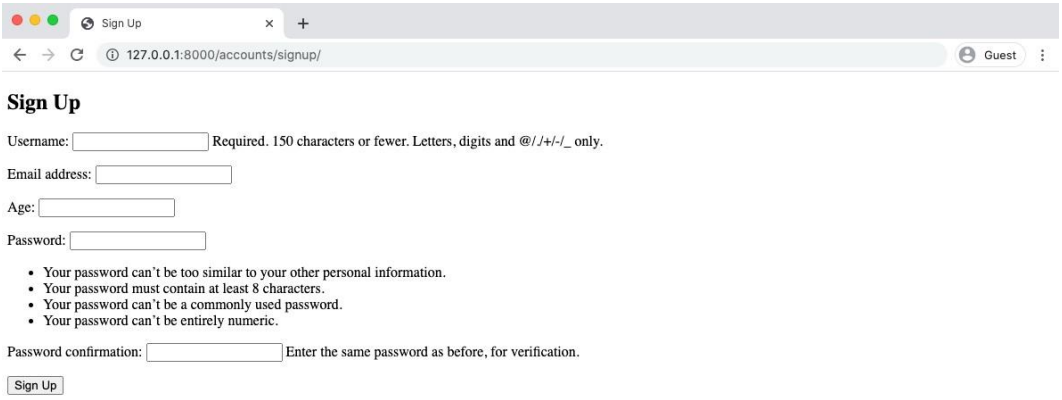
Users in the Admin

كل شيء يعمل ولكن قد تلاحظ أنه لا يوجد "عنوان بريد إلكتروني" لمستخدم الاختبار لدينا. لماذا هذا؟ حسنا ، لا تحتوي صفحة التسجيل الخاصة بنا على حقل بريد إلكتروني لأنه لم يتم تضمينه في الحسابات / **forms.py**. هذه نقطة مهمة: لمجرد أن نموذج المستخدم يحتوي على حقل ، فلن يتم تضمينه في نموذج الاشتراك المخصص ما لم تتم إضافته صراحة. دعونا نفعل ذلك الآن. حاليا، في الحسابات/ **forms.py** ضمن الحقول، نستخدم **Meta.fields** ، الذي يعرض فقط الإعدادات الافتراضية لاسم المستخدم/العمر/كلمة المرور. ولكن يمكننا أيضا تعيين الحقول التي نريد عرضها بشكل صريح ، لذا دعنا نقوم بتحديثه لطلب اسم مستخدم / بريد إلكتروني / عمر / كلمة مرور عن طريق تعيينه على ("اسم المستخدم" ، "البريد الإلكتروني" ، "العمر"). لسنا بحاجة إلى تضمين حقول كلمة المرور لأنها مطلوبة! يمكن تكوين جميع الحقول الأخرى كيفما نختار .

Code

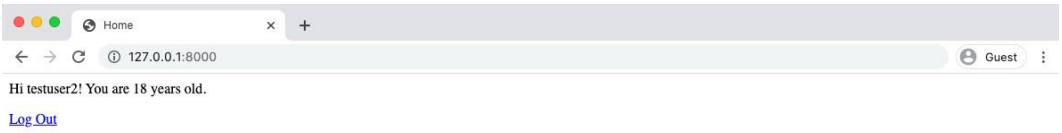
```
# accounts/forms.py from django import forms from django.contrib.auth.forms import
UserCreationForm, UserChangeForm from .models import CustomUser


class CustomUserCreationForm(UserCreationForm):

    class Meta(UserCreationForm):
        model = CustomUser fields = ('username', 'email', 'age',) # new


class CustomUserChangeForm(UserChangeForm):

    class Meta:
        model = CustomUser
        fields = ('username', 'email', 'age',) # new
```

Now if you try `http://127.0.0.1:8000/accounts/signup/` again you can see the additional "Email address" field is there.



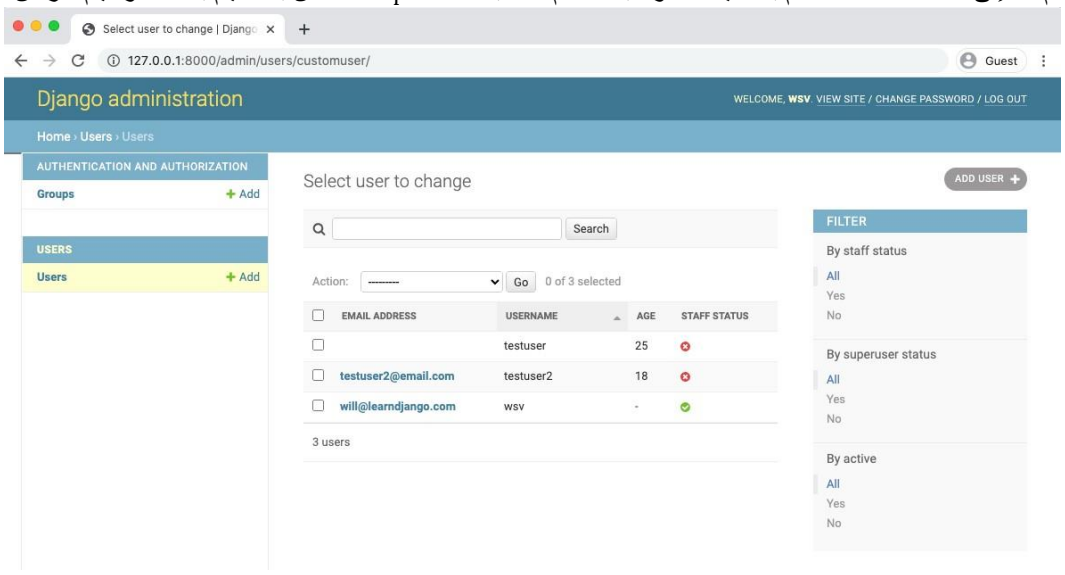
New sign up page

Sign up with a new user account. I've named mine `testuser2` with an age of `18` and an email address of `testuser2@email.com`. Continue to log in and you'll see a personalized greeting on the homepage.



testuser2 homepage greeting

ثم عد إلى صفحة Admin - قم بتسجيل الدخول باستخدام حساب superuser الخاص بنا للقيام بذلك – وسيتم عرض جميع المستخدمين الثلاثة.



Three users in the Admin

يتطلب تدفق مصادقة المستخدم في **Django** القليل من الإعداد ولكن يجب أن تبدأ في رؤية أنه يوفر لنا أيضا مرونة لا تصدق لتكوين التسجيل وتسجيل الدخول بالطريقة التي نريدها بالضبط.

الخلاصة

حتى الآن ، يحتوي تطبيق Newspaper الخاص بنا على نموذج مستخدم مخصص وصفحات تسجيل الدخول وتسجيل الدخول وتسجيل الخروج. ولكن ربما لاحظت أن موقعنا لا يبدو جيدا جدا. في الفصل التالي ، سنضيف Bootstrap للتصميم وإنشاء تطبيق .pages