# به نام خدا

## گزارش فنی **Bookstore Project**

زهرا خلیفه زاده – مکتب 51

## Chapter 3

هدف ساخت پروژه کتاب فروشی آنلاین است. یک مدل یوزر کاستوم اضافه میکنیم. در این پروژه به جای db.sqlite3 که پیش فرض جنگو است، از PostgreSQL برای دیتابیس استفاده شد و تنظیمات آن در settings.py اضافه میشود.

**مدل کاربر کاستوم:**

قبلا روش پیشنهاد شده برای مدل کاربر سفارشی، افزودن OneToOneField بود. الان میتوان از گزینه های پیاده سازی شده درخود جنگو استفاده کرد. میتوان از AbstractUser یا AbstractBaseUser استفاده کرد تا مجوزها و دسترسی های کاربران را مشخص کرد. در این پروژه از AbstractUser که ساده تر است استفاده میشود.

### تفاوت AbstractUser و AbstractBaseUser :

در جنگو به صورت پیش فرض مدل user بیس از django.contrib.auth.models ایمپورت میشود. در پروژه ها با استفاده از AbstractBaseUser و AbstractUser این اجازه را داریم که مدل user جنگو را override کنیم و یک کلاس یوزر کاستوم در فایل مدل خود بسازید. هر دو از آدرس django.contrib.auth.models در جنگو ایمپورت میشوند.

**AbstractBaseUser** همان AbstractUser است با این تفاوت که فقط اطلاعات بیس را در اختیار شما قرار میدهد. AbstractBaseUser فقط آن اطلاعاتی که برای authenticat یا اعتبارسنجی کاربر لازم دارید (مثل پسورد) را به شما میدهد. چند متد دیفالت هم دارد مثل تغییر پسورد، چک کردن پسورد و غیره. تمام این متدها هم مربوط به authenticat هستند.

**AbstractUser** اطلاعات بیشتری از کاربرها در اختیار شما قرار میدهد. AbstractUser از AbstractBaseUser ارث بری میکند. پس علاوه بر فیلدها و متدهای AbstractBaseUser یک سری امکانات و اطلاعات اضافه تری هم دارد. مثلا یوزرنیم، فرست نیم و یک سری متد اضافی مثل get_full_name را دارد. AbstractUser از PermissionsMixin ها هم استفاده میکند. PermissionsMixin یک سری کلاس هستند که با اختیاراتی که یوزر دارد (مثل گروه ها) سرو کار دارند.

پس اگر برای یوزر کاستوم خود از AbstractBaseUser ارث بری کنید باید فیلد های اضافی مثل username را که میخواهید خودتان به کلاس یوزر کاستومتان اضافه کنید. ولی اگر از AbstractUser ارث بری میگردید دیگر نیاز به نوشتن این فیلدها نیست. چون خود AbstractUser آنها را دارد.

### ساخت مدل CustomUser:

ابتدا یک اپ با نام accounts میسازیم. در این اپ یک مدل CustomUser میسازیم که از AbstractUser ارث بری میکند. با اینحال میتوان به این مدل عملکرد جدید اضافه کرد و یا آن را طبق نیاز پروژه تغییر داد.

اپ را در settings.py/INSTALLED_APPS اضافه میکنیم. همچنین برای اینکه بتوان به جای مدل کاربر پیش فرض از CustomUser استفاده کنیم باید در ستینک پروژه AUTH_USER_MODEL = 'accounts.CustomUser قرار میدهیم. سپس migrations و migrate انجام میدهیم تا دیتابیس initialize شود.

### ساخت فرم CustomUser :

یک مدل کاربر در جنگو میتواند کارهای ویرایش و ایجاد را انجام دهد. پس باید فرم های داخلی را آپدیت کنیم تا به جای یوزر جنگو از CustomUser استفاده کند. پس یک forms.py در اپ میسازیم. دو کلاس فرم برای ویرایش و ایجاد میسازیم که از UserCreationForm و UserChangeForm ارث بری میکند. مدل این کلاس های فرمی که ساختیم get_user_model است که به AUTH_USER_MODEL که در ستینک اضافه کردیم اشاره دارد. و دیگر نیازی به وارد کردن مستقیم CustomUser در کل پروژه نیست. میتوان در هر کلاس فیلدهای مورد نیاز برای نمایش را با fields مشخص کرد.

### ادمین CustomUser:

حالا باید فایل admin.py اپ مان را آپدیت کنیم. یوزر ادمین موجود را به CustomUserAdmin گسترش میدهیم. میتوان فرم ها، فیلدهای دلخواه و نحوه نمایش را برای صفحه ادمین مشخص کرد. سپس createsuperuser را انجام میدهیم تا بتوانیم وارد پنل ادمین شویم.

# Chapter 4

### templates :

میخواهیم یک صفحه اصلی برای پروژه خود بسازیم. فعلا این صفحه با دیتابیس در ارتباط نیست. بعدا یک صفحه ایجاد میکنیم تا بتوانیم کتاب های ثبت شده را ببینیم. پس الان یک اپ جدید به نام pages ایجاد میکنیم. آن را به INSTALLED_APPS اضافه میکنیم. بعد باید یک پوشه templates در پروژه ساخت و مسیر آن را به settings.py/TEMPLATES اضافه کنیم. در پوشه تمپلت دو فایل _base.html وhome.html را ایجاد میکنیم.

فایل base.html توسط فایل های دیگر به ارث برده میشود و برخی برنامه نویسان به سلیقه خود فایل هایی که به ارث برده میشوند را با یک _ نمایش میدهند. در این فایل تک بلاک title و content قرار دارد. به جای نام content میتوان هر اسم دلخواهی را به کار برد ولی بهتر است مثل سایر کاربران جنگو از همین نام مشترک استفاده کرد. محتوای اصلی پروژه در این بلاک قرار میگیرد.

### URLs and Views:

در جنگو هر webpage ای برای برقراری ارتباط با templates به یک urls.py و views.py نیاز دارد. در config/urls.py/urlpatterns اصلی پروژه باید مسیر path('', include('pages.urls')) را اضافه کنیم. سپس pages/urls.py را ایجاد میکنیم. در pages/views.py یک کلاس HomePageView که از TemplateView ارث بری میکند را ایجاد میکنیم و template_name آن را برابر با صفحه html مورد نظر خود قرار میدهیم. حالا این کلاس را به صورت HomePageView.as_view() به مسیرها اضافه میکنیم. یک نام هم برای آن مشخص میکنیم که بعدا در مسیر دهی برای پروژه بتوان از آن استفاده کرد.

با ران کردن پروژه میتوانید در صفحه اول خود این صفحه home ای را که خودتان ساختید به جای صفحه پیش فرض جنگو مشاهده کنید.

## Chapter 5: User Registration

بخش اصلی هر سایت registration است. در این فصل عملکرد ثبت نام، لاگین و لاگ اوت پیاده سازی میشود. برای لاگین و لاگ اوت خود جنگو ویوهای لازم را دارد ولی برای ساین آپ یا ثبت نام راه حل built-in ندارد.

**Auth App**: برای استفاده از آن باید در INSTALLED_APPS ستینک اضافه شود. این اپ با ساخت پروژه جنگو به صورت خودکار اضافه شده است. وقتی برای بار اول پروژه را migrate میکنیم همه این اپ ها به دیتابیس اضافه میشود. به یاد داشته باشید که در فصل قبل به جای یوزر جنگو از یوزر کاستوم با AUTH_USER_MODEL استفاده کردیم.

**Auth URLs and Views**: برای استفاده از اپ auth باید آن را در urls.py پروژه اضافه کرد. با /accounts به آن دسترسی داریم و از django.contrib.auth.urls اینکلود شده است که شامل لاگین، لاگ اوت و ... است.

**Homepage**: حالا باید صفحه home را آپدیت کنیم تا فقط کاربری که لاگین است دسترسی ها را داشته باشد. باید از تگ های if/else استفده کنیم. اگر کاربر لاگین باشد یا authenticated به آن پیام خوشامد با نام کاربر گفته میشود واگر لاگین نباشد میگوید شما لاگین نیستی و به صفحه لاگین فرستاده میشود. در این جا خود جنگو url برای loginوlogout را دارد که با درج نام این تمپلت به آن لینک میشود.

**Django Source Code**: یوزر و متغیرهای مربوط به آن چطور در تمپلت در دسترس هستند؟ جنگو به صورت خودکار context تمپلت را استفاده میکند. هر تمپلت با فایل view.py لود میشود. میتوانیم با تگ تمپلت از ویژگی های یوزر استفاده کنیم. با استفاده از is_authenticated در تمپلت چک میکنیم که یوزر لاگین هست یا نه. اگر لاگین بود ایمیل یوزر را برمیگرداند. با مسیر accounts/login/ میتوانیم به صفحه لاگین وارد شویم. این مسیرهای پیش فرض در django.contrib.auth جنگو وجود دارند. در django/contrib/auth/urls.py جنگو میتوان مسیردهی را بر اساس متدهای ویوی خود جنگو مشاهده کرد. برای login از LoginView استفاده کرده است.

```
from django.contrib.auth import views

urlpatterns = [path('login/', views.LoginView.as_view(), name='login'),]
```

**Log In**: برای لاگین شدن باید templates/registration/login.html خودمان را بسازیم. در تمپلت لاگین یک فرم با متد post داریم. به هر فرمی باید تگ {% csrf_token %} را اضافه کنید. as_p() فرم را به صورت تگ p نمایش میدهد. با رفرش صفحه وارد صفحه لاگین میشوید ولی با زدن لاگین با خطا مواجه میشوید.

**Redirects**: با زدن لاگین با خطا مواجه میشود چون جنکو شما را به /accounts/profile ریدایرکت میکند. این صفحه ای است که هنوز ایجاد نشده است. برای اینکه بعد از لاگین به صفحهhome ریدایرکت شوید LOGIN_REDIRECT_URL = 'home' را به ستینگ پروژه اضافه میکینم.

**Log Out**: برای اضافه کردن گزینه logout به صفحه home میتوان LogoutView نوشت. ولی ما از LOGOUT_REDIRECT_URL = 'home' در ستینگ پروژه استفاده میکنیم. و بعد در home.html باید لینک برای ریدایرکت شدن اضافه کنیم. نام ویو آن در جنگو login است که در تگ لینک اضافه میکنیم.

**Sign Up**: پیاده سازی صفحه signup برای ثبت نام چند مرحله استاندارد دارد. باید accounts/urls.py را ایجاد کرد. config/urls.py را برای اپ accounts آپدیت کرد. سپس ویو SignupPageView را اضافه کنید. تمپلت signup.html را بسازید. . بعد home.html را برای نمایش صفحه ساین آپ آپدیت کنید. در مدل این اپ ما مدل CustomUser را داریم که از AbstractUser ارث بری کرده است. در فایل ویو باید متد SignupPageView را تعریف کنیم. سپس url را برای این متد با نام signup تعریف کنیم. config/urls.py را آپدیت میکنیم و accounts.urls را به آن اضافه میکنیم. حالا ویو SignupPageView را میسازیم که به CustomUserCreationForm اشاره دارد. و یک success_url که به صفحه login برمیگردد. یعنی بعد از سابمیت کردن فرم کاربر به صفحه لاگین ریدایرکت میشود. تمپلت این ویو signup.html است. سپس به home.html لینک این صفحه را اضافه میکنیم. صفحه را رفرش کنید و یک یوزر جدید ثبت نام کنید. بعد از ساین آپ به Log In page ریدایرکت میشوید. با این یوزر جدید لاگین شوید و به homepage ریدایرکت میشوید.

## Chapter 6: Static Assets

استاتیک ها شمل فایل های css ، جاوا اسکریپت و image ها هستند. جنگو امکاناتی برای استفاده از این فایل ها در اختیار ما قرار میدهد. میتوانیم از بوت استرپ برای بهبود استایل استفاده کنیم.

**staticfiles app**: جنگو با اپ staticfiles کل فایل های استاتیک را مدیریت میکند. آنها را در یک مکان واحد جمع آوری میکند. پیکربندی آن را باید در settings.py اضافه کنیم.

**STATIC_URL**: این مسیر در ستینک وجود دارد. این URL ای را مشخص میکند که جهت ارجاع به فایل های استاتیک از آن استفاده میکنیم.

**STATICFILES_DIRS**: محل فایل های استاتیک را در لوکال مشخص میکند. با [] میتوان چند مسیر را مشخص کرد.

**STATIC_ROOT**: محل فایل های استاتیک برای تولید را مشخص میکند. بعد از ساخت پروژه جنگو دستور collectstatic را میزنیم تا به طور خودکار تمام فایل های استاتیک در کل پروژه در این پوشه جمع شوند.

**STATICFILES_FINDERS**: به جنگو میگوید چطور دنبال فایل های استاتیک پروژه بگردد. به صورت خودکار تنظیم شده است. میتوان به صورت اختیاری آن را تعریف کنیم. در این تنظیمات FileSystemFinder به STATICFILES_DIRS نگاه میکند. و AppDirectoriesFinder به دنبال هر فهرست static است که داخل هر اپ وجود دارد.

**Static Directory**: حتی اگربه یک دایرکتوری ثابت برای فایل ها اشاره میکنیم باید مسیر استاتیک آن را با زیرشاخه هایی css، js و image ایجاد کنیم. مثلا اگر یک فایل css اضافه کنید باید در تمپلت مربوطه آن را با {% load static %} در بالای صفحه لود کنید. تگ تمپلت استاتیک به STATIC_URL اشاره دارد. اگر این css را در base.html اضافه کنید به کل صفحه اضافه میشود.

**Images**: برای اضافه کردن عکس در یک تمپلت ابتدا باید عکس را در مسیرimage قرار دهید. بعد با تگ img در فایل تمپلت آن را لود کنید. مثل تمام فایل های استاتیک باید در ابتدای صفحه استاتیک لود شود. برای تغییر سایز عکس از یک css استفاده کنید.

**JavaScript**: برای اضافه کردن یک فایل js باید base.js را در پوشه استاتیک مربوطه ایجاد کنید. سپس آن را به base.html_ اضافه کنید. جاوا اسکریپت باید به انتهای فایل اضافه شود. پس بعد از htmlو css لود میشود.

**Collectstatic**: در این مرحله باید دستور collectstatic را اجرا کنیم تا تمام فایل های استاتیک پروژه در یک جا جمع شوند. حالا یک پوشه با نام staticfiles ایجاد شده است. همان نامی که در STATIC_ROOT مشخص کردیم.

**Bootstrap**: به جای نوشتن css کاستوم میتوان از بوت استرپ استفاده کرد. فایل های بوت استرب باید در بالای صفحه لود شوند و فایل های جی کوئری در پایین. لینک فایل های بوت استرپ را در ابتدای صفحه میتوان اضافه کرد. برای اجرای آنها باید به اینترنت متصل بود. در غیر این صورت میتوان فایل را دانلود و به صورت لوکال مسیر داد. یک نوبار اضافه کنید که لینک به صفحات دیگر داشته باشد.

**About Page**: یکی از لینک هایی که در نوبار مرحله قبل با بوت استرب اضافه کردیم صفحه about است. پس about.html را به تمپلت ها اضافه میکنیم. برای دیدن آن باید ویو TemplateView را ایجاد کنیم. و مسیر /about را در url با توجه به متد ویو اضافه کنید. با رفرش صفحه میبینید که استایلی که از صفحه base.html_ گرفته بود اعمال شده است.

**Django Crispy Forms**: به روز رسانی مربوط به فرم ها است. برای استفاده باید با دستور install django-crispy-forms==1.9.2 آن را نصب کنید. سپس crispy_forms را به INSTALLED_APPS اضافه کنید. همچنین CRISPY_TEMPLATE_PACK = 'bootstrap4' در ستینک اضافه کنید. برای استفاده از آن در فرم ها باید تگ {{ form|crispy}} را به جای {{ form.as_p }} قرار دهید. قبل از هر چیز {% load crispy_forms_tags %} را لود کنید.

# Chapter 7: Advanced User Registration

در فصل 5 ثبت نام استاندارد کاربر را پیاده سازی کردیم. اما حرفه‌ای‌تر آن است که ثبت نام پیشرفته کاربر را انجام دهیم. یک سری چیزها را سفارشی کنیم. به صورت پیش فرض برای هر ثبت‌نام به نام کاربری، ایمیل و پسورد نیاز داریم. هر بخشی از احرازهویت یا authentication میتواند در صورت نیاز سفارشی شود.

یکی از عوامل مهم در پروژه ها social authentication است، یعنی ثبت نام و لاگین از طریق خدمات شخص ثالث مانند گوگل و فیسبوک انجام شود. ثبت نام کاربر یک کار پیچیده است و نباید اشتباه امنیتی انجام دهیم. به همین دلیل بسیاری از توسعه دهندگان به djangoallauth شخص ثالث در جنگو تکیه میکنند. اضافه کردن هر یک از این پکیج ها باید با احتیاط انجام شود زیرا یک وابستگی را به بخش تکنیکال اضافه میکند. هر پکیج باید آپدیت شده و تست شده باشد. در جنگو django-allauth هر دو مورد را دارد. با کمی هزینه کاستومایز کردن آسان میشود.

**django-allauth**: باید django-allauth را نصب کنید. سپس با اضافه کردن allauth.account به settings.py/INSTALLED_APPS تنظیمات پروژه را را آپدیت میکنیم. فریم ورک سایت‌های جنگو یک ویژگی قدرتمند است که به پروژه جنگو اجازه کنترل چندین سایت را میدهد. ما فقط یک سایت در پروژه خود داریم. پس SITE_ID را برابر 1 میگذاریم. اگر سایت دوم را اضافه کنیم ID برابر 2 میشود. به همین ترتیب شماره ID افزایش می یابد.

**AUTHENTICATION_BACKENDS**: فایل settings.py که در ابتدای هر پروژه جدید جنگو ایجاد میشود دارای تعدادی تنظیمات است. یک سری تنظیمات را خودمان میتوانیم اضافه کنیم، مثل اضافه کردن اپ ها در لیست اپ های نصب شده. یک سری تنظیمات هم قابل دیدن نیست.

میتوان AUTHENTICATION_BACKENDS را به فایل ستینگ اضافه کرد. جنگو در زمان تلاش برای احراز هویت یا authenticate یک کاربر ازآن استفاده میکند. همچنین باید گزینه های authentication خاص را برای django-allauth اضافه کنیم، که به ما اجازه ورود به سیستم از طریق e-mail را میدهد. پس تنظیمات آن را به settings.py اضافه میکنیم.

**EMAIL_BACKEND**: به طور ضمنی configuration مربوط به EMAIL_BACKEND را تعریف میکنیم. جنگو به طور پیش فرض به دنبال یک سرور SMTP برای ارسال ایمیل است. با ثبت نام موفق یک کاربر، django-allauth یک ایمیل ارسال میکند. میتوان آن را بعدا کاستومایز کرد. اگر پیکربندی سرور SMTP را انجام دهیم با خطا مواجه میشویم. در حال حاضر یک را ه حل این است که جنگو ایمیل را به command line کنسول ارسال کند. بنابراین میتوانیم به جای smtp از console در پیاده سازی استفاده کنیم و EMAIL_BACKEND را به ستیک اضافه میکنیم.

**ACCOUNT_LOGOUT_REDIRECT**: در فصل قبل برای ACCOUNT_LOGOUT_REDIRECT تنظیماتی را انجام دادیم که به صورت پیش فرض به homepage ریدایرکت شود. مسئله اینجاست که django-allauth دیگر LOGOUT_REDIRECT_URL را override میکند. اما هر دو به homepage اشاره میکنند. این تغییر ممکن است آشکار نباشد. ممکن است بعدا ما نخواهیم که همیشه به homepage ریدایرکت یا هدایت شویم، پس میتوانیم ریدایرکت logout را به صورت واضح مشخص کنیم.

باتوجه به تغییرات زیادی که در config/settings.py انجام دادیم باید migrate را اجرا کنیم.

**URLs**: حالا که از django-allauth استفاده کردیم، باید auth app URLs را با اپ allauth عوض کنیم. همچنان از accounts/ URL path استفاده میکنیم ولی از تمپلت های allauth برای sign up استفاده خواهیم کرد. میتوانیم URL path مربوط به اپ accounts را حذف کنیم. تا قبل از اضافه کردن django-allauth، برای صفحه sign up دستی از accounts/urls.py و accounts/views.py استفاده میکردیم. الان دیگر استفاده نمیشوند و میتوان حذفشان کرد.

**Templates**: در اپ auth جنگو به دنبال مسیر templates/registration میگردد. اما در allauth به دنبال مسیر templates/account میگردد. پس ما دایرکتوری templates/account را ایجاد و فایل های login.html و signup.html را در آن قرار میدهیم. میتوان templates/registration را حذف کرد چون دیگر به آن نیازی نداریم. در گام آخر باید templates/_base.html و templates/home.html را برای django-allauth's URL به جای Django آپدیت کنیم. با اضافه کردن یک پسوند account به logout و signup این کار را انجام میدهیم.

**Log In**: صفحه homepage را رفرش و login کنید. صفحه لاگین را نمایش میدهد. دقت کنید که ما accounts/urls.py و accounts/views.py نداریم. ولی به صفحه لاگین هدایت شدیم. به صفحه لاگین دقت کنید که باکس Remember Me به آن اضافه شده است. این پیکربندی را django-allauth ارائه میدهد. از کاربر میپرسد که آیا میخواهد session اش به خاطر سپرده شود تا دیگر مجبور به لاگین دوباره نباشد یا نه. اگر نخواهید این باکس اضافه شود باید در ستینگ پروژه ACCOUNT_SESSION_REMEMBER را برابرTRUE قرار دهید. اگر با اکانت superuser لاگین کنید، به homepage هدایت میشوید. دکمه logout را بزنید. به جای خروج مستقیم از سایت django-allauth یک صفحه واسط Log Out دارد که میتوان آن را برای بقیه پروژه کاستومایز کرد.

**Log Out**: تمپلت پیش فرض Log Out را با ایجاد account/logout.html میتوانید override کنید.

**Sign Up**: لینک Sign Up در نوبار بالای صفحه دارای استایل Bootstrap و django-crispy-forms است. در این فرم میبینید که دوبار رمز عبور را باید وارد کنید. یک سفارشی سازی در django-allauth این است که میتوانید با ACCOUNT_SIGNUP_PASSWORD_ENTER_TWICE = False فقط یک بار پسورد را درخواست کنید. بعدا گزینه های تغییر و ریست پسورد را پیاده سازی میکنیم. پس این احتمال وجود ندارد که کاربری که رمز عبور را اشتباه وارد میکند در اکانت خود locked out بماند.

یک کاربر جدید با نام testuser1 و ایمیل testuser1@email.com با پسورد testpass123 ایجاد کنید.بعد از submitبه homepage هدایت میشوید. حالا به یاد بیاورید که چطور email را برای خروجی کنسول در EMAIL_BACKEND مشخص کردیم. پس django-allauth به طور خودکار یک ایمیل بعد از registration ارسال میکند که در کنسول قابل مشاهده است. بعدا یک سرویس email مناسب برای ارسال به کاربران واقعی را پیکربندی میکنیم.

**Admin**: با اکانت superuser وارد صفحه ادمین شوید و میبینید که django-allauth درگیر شده است. دو بخش جدید Accounts و Sites با تنظیمات جدیدی که انجام دادیم اضافه شده است. در بخش بعدی پیکربندی ایمیل را برای نام دامنه و نام نمایشی انجام میدهیم.

**Email Only Login**: زمان آن رسیده که از لیست گسترده django-allauth استفاده کنیم تا برای login به جای username فقط از email استفاده کنیم. به چند تغییر نیاز داریم. در ستینگ پروژه باید ACCOUNT_USERNAME_REQUIRED = False و ACCOUNT_EMAIL_REQUIRED = True قرار دهیم. یعنی نام کاربری required نیست ولی ایمیل required است. سپس به ایمیل unique و متد authentication انتخابی نیاز داریم.

حالا logout کنید و با Sign Up یک کاربر جدید با ایمیل testuser2@email.com و پسورد testpass123 ایجاد کنید. پس از ثبت نام موفق به homepage هدایت میشوید. حالا وارد پنل ادمین شوید تا تغییرات را ببینید. مشاهده میکنید که django-allauth به صورت خودکار یک نام کاربری بر اساس ایمیل برای شما مشخص کرده است. دلیلش این است که مدل CustomUser هنوز یک فیلد username دارد. همچنین از نوع AbstractUser است.

برای حذف username از مدل کاربر سفارشی، باید از AbstractBaseUser استفاده کنید. تفاوت AbstractBaseUser و AbstractUser در فصل3 توضیح داده شده است.

سوال: اگر با ایمیل testuser2@example.com یک کاربر جدید ثبت نام کنیم، برای username مشکل پیش می آید؟ خیر.

اگر username مربوط به دو ایمیل مشابه باشد ، به صورت خودکار django-allauth یک string دورقمی random به نام کاربری اضافه میکند.

## Chapter 9: Email

در این فصل پیکربندی ایمیل و قابلیت تغییر پسورد و reset پسورد را اضافه میکنیم. در حال حاضر ایمیل ها برای کاربران ارسال نمیشود و فقط در کنسول ارسال میشود. با ثبت نام در یک سرویس ایمیل شخص ثالث، بدست آوردن API keys و آپدیت settings.py میتوانیم به صورت واقعی به کاربران ایمیل ارسال کنیم.

**Custom Confirmation Emails**: برای مرور یک کاربر با ایمیل testuser3@email.com و پسورد testpass123 با Sign Up ایجاد کنید. پس از ثبت نام میبنید که به homepage هدایت میشوید و ایمیل ارسالی را در کنسول مشاهده میکنید.

برای سفارشی کردن این ایمیل، ابتدا باید تمپلت های موجود را پیدا کنیم. باید فایل email_confirmation_subject.txt برای subject ایمیل و email_confirmation_message.txt برای body ایمیل را ایجاد کنیم. برای آپدیت این دو فایل باید با override مشابه ساختار django-allauth با ایجاد مجدد دوفایل در templates/account/email آنها را آپدیت کنیم. سپس با تگ ها و بلاک های مورد نظر این دو فایل را مینویسیم.

در متن این دو فایل ایمیل پیش فرض ارسال شده به سایت ما به example.com اشاره دارد که با عنوان {{ site_name }} نمایش داده شده است. در پنل ادمین در بخش sites که توسط django-allauth استفاده میشود وجود دارد. به این بخش بروید و فیلدهای آن را ببینید. فیلدهای Domain Name که نام کامل دامنه برای سایت و Display Name که نام نمایش داده شده برای سایت است را مشاهده میکنید. هر دو قابل ادیت شدن هستند و میتوان نام دلخواه را جایگزین کرد.

میتوانیم بخش email را کمی بیشتر سفارشی کنیم. در فایل email_confirmation_message.txt کلمه Hello را به Hi تغییر دهید. ایمیلی که ارسال میشد به صورت دیفالت از webmaster@localhost بود. میتوانیم با DEFAULT_FROM_EMAIL در ستینگ آن را به ایمیل خود تغییر دهیم. بعد از این تغییرات یک یوزر جدید به نام testuser4@email.com و پسورد testpass123 ایجاد کنید. بعد از ثبت نام به homepage ریدایرکت میشوید . در کنسول میتوانید message و دامنه جدید djangobookstore.com و همچنین ایمیل پیش فرضی که در ستینگ تغییر دادید را ببینید.

**Email Confirmation Page**: پس از ثبت نام و مشاهده ایمیل ارسالی در کنسول، روی لینک URL یونیک ایمیل که به صفحه تایید ایمیل میرود کلیک کنید. میتوانید ظاهر آن را مطابق بقیه سایت تغییر دهید. قالب خود را با اضافه کردن templates/account/email_confirm.html میتوانید تغییر دهید. در این فایل base.html_ را extends کرده و از Bootstrap برای دکمه استفاده کنید. صفحه را رفرش کنید و قالب جدید را مشاهده کنید که ظاهر زیبایی دارد.

**Password Reset and Password Change**: جنگو و django-allauth دارای ویژگی هایی برای کاربر، مانند تغییر پسورد در هنگام لاگین و ریست پسورد در صورت فراموشی پسورد است. تمپلت ها و messages ایمیل در کد منبع django-allauth است.

http://127.0.0.1:8000/accounts/password/reset/
http://127.0.0.1:8000/accounts/password/change/

**Email Service**: به ایمیل هایی که تا اینجا پیکربندی کردیم Transactional Emails میگویند. این ایمیل ها براساس یک نوع عملکرد user هستند. این نوع ایمیل ها بر خلاف Marketing Emails، مثلا خبرنامه ماهانه است.

ارائه دهندگان transactional email مثل SendGrid، MailGun و Amazon سرویس های ساده ایمیل در دسترس هستند. در یکی از این سرویس های ایمیل ثبت نام کنید. بین Web API و SMTP میتوانید انتخاب کنید. پیکربندی SMTP ساده تر است، با آن شروع کنید. پس از ثبت نام و دریافت نام کاربری از سرویس ارائه دهنده ایمیل، یک سری تنظیمات در جنگو باید انجام دهید تا از این سرویس برای ارسال ایمیل واقعی به جای کنسول استفاده کنید. ابتدا باید EMAIL_BACKEND را در ستینگ پروژه به SMTP تغییر دهید. همچنین باید EMAIL_HOST_USER, EMAIL_HOST, EMAIL_HOST_PASSWORD, EMAIL_PORT, EMAIL_USE_TLS را بر اساس دستورالعمل سرویس ارائه دهنده پیکربندی کنید. در این کتاب از همان کنسول استفاده شده است و این تغییرات را اعمال نکردیم.

نکته: برای ارسال سرویس ایمیل واقعی از gmail استفاده کردم:

```
# use gmail for email service
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend' # new
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = 'zahra.kh2005@gmail.com'
EMAIL_HOST_PASSWORD = 'wkofzkrxnwjzbymk'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
```