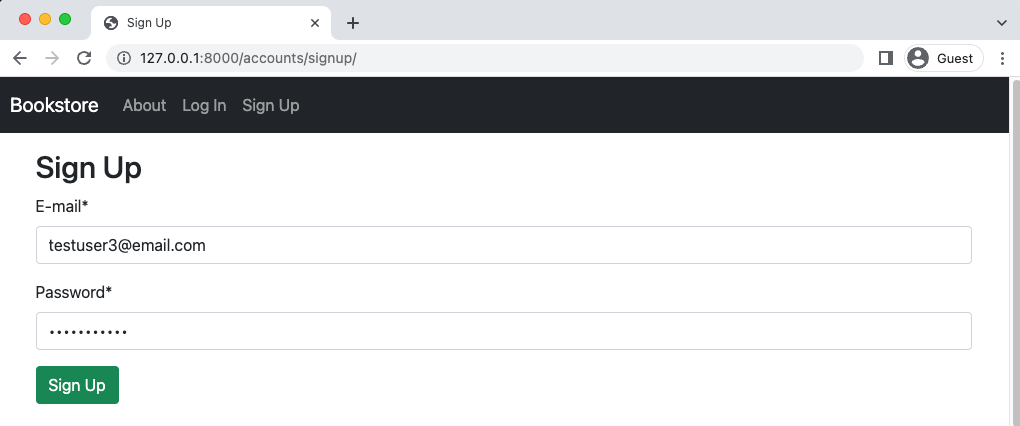
**Chapter 10: Email**

In this chapter we will fully configure email and add password change and password reset functionality. Currently emails are not actually sent to users. They are simply outputted to our command line console. We’ll change that by signing up for a third-party email service, obtaining API keys, and updating our django\_project/settings.py file. Django takes care of the rest.

So far all of our work–custom user model, pages app, static assets, authentication with allauth, and environment variables–could apply to almost any new project. After this chapter we will start building out the Bookstore site itself as opposed to foundational steps.

**Custom Confirmation Emails**

Let’s sign up for a new user account to review the current user registration flow. Then we’ll customize it. Make sure you are logged out and then navigate to the Sign Up page. I’ve chosen to use testuser3@email.com and testpass123 as the password.

  
**testuser3 Sign Up**

Upon submission we are redirected to the homepage with a custom greeting and an email is sent to us within the command line console. You can see this by checking the logs with docker-compose logs. Here is a snippet of the much longer output.

docker-compose logs

...

Hello from example.com!

| You're receiving this e-mail because user testuser3 ...

...

To customize this email we first need to find the existing templates. Navigate over to the djangoallauth source code on Github and perform a search with a portion of the generated text. For example, “You’re receiving this e-mail.” This leads to the discovery of a email\_confirmation\_message.txt file located within django-/allauth/templates/account/email. If you look at this directory’s content there is also a subject line file, email\_confirmation\_message.txt that we can and will change.

To customize these files we’ll override them by recreating the same structure of django-allauth in our project. That means creating an email directory within the templates/account directory.

mkdir templates/account/email

And then in the text editor create the two new files:

• templates/account/email/email\_confirmation\_subject.txt

• templates/account/email/email\_confirmation\_message.txt

Let’s start with the subject line since it’s the shorter of the two. Here is the default text from django-allauth.

email\_confirmation\_subject.txt

{% load i18n %}

{% autoescape off %}

{% blocktrans %}Please Confirm Your E-mail Address{% endblocktrans %}

{% endautoescape %}

The first line, {% load i18n %}, is to support Django’s internationalization functionality which supports multiple languages. Then comes the Django template tag for autoescape. By default it is “on” and protects against security issues like cross site scripting. But since we can trust the content of the text here, it is turned off.

Finally, we come to our text itself which is wrapped in blocktrans template tags to support translations. Let’s change the text from “E-mail Address” to “Sign Up” to demonstrate that we can.

email\_confirmation\_subject.txt

{% load i18n %}

{% autoescape off %}

{% blocktrans %}Confirm Your Sign Up{% endblocktrans %}

{% endautoescape %}

Now turn to the email confirmation message itself. Here is the current default:

email\_confirmation\_message.txt

{% extends "account/email/base\_message.txt" %}

{% load account %}

{% load i18n %}

{% block content %}{% autoescape off %}{% user\_display user as user\_display %}\

{% blocktrans with site\_name=current\_site.name site\_domain=current\_site.domain %}\

You're receiving this e-mail because user {{ user\_display }} has given your\

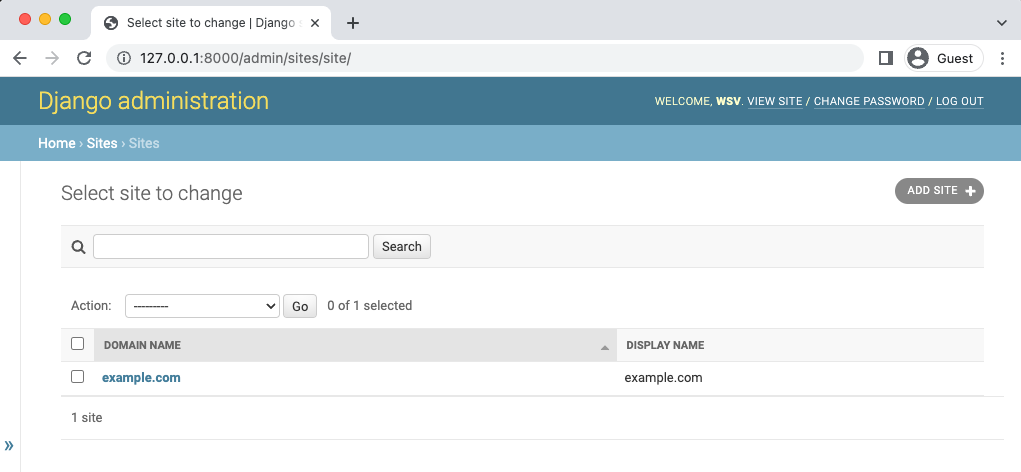
e-mail address to register an account on {{ site\_domain }}.

To confirm this is correct, go to {{ activate\_url }}\

{% endblocktrans %}{% endautoescape %}{% endblock %}

|  |
| --- |
| **Note:** that backslashes \ are included for formatting but are not necessary in the raw code. In other words, you can remove them from the code below–and other code examples–as needed. |

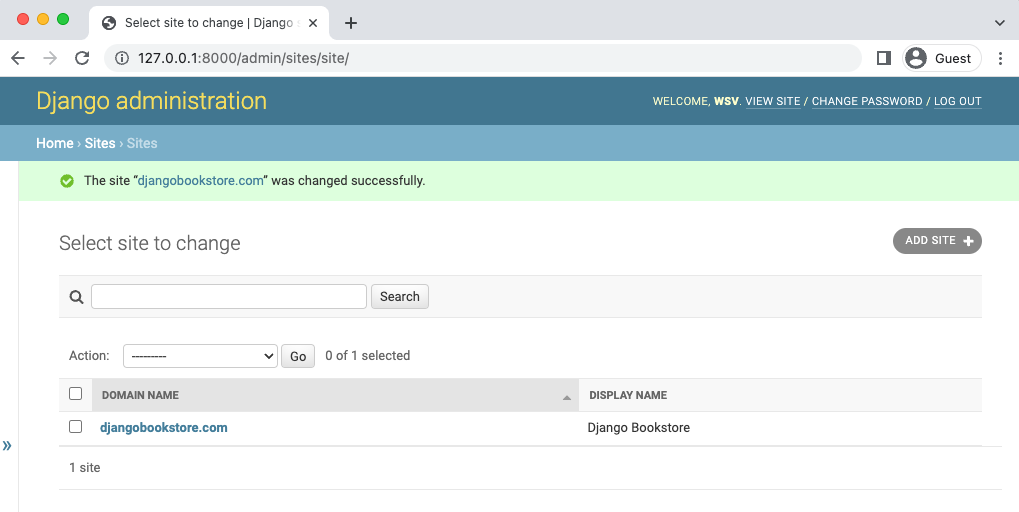
You probably noticed that the default email sent referred to our site as example.com which is displayed here as {{ site\_name }}. Where does that come from? The answer is the sites section of the Django admin, which is used by django-allauth. So head to the admin at http://127.0.0.1:8000/admin/ and click on the Sites link on the homepage.



Admin Sites

There is a “Domain Name” and a “Display Name” here. Click on example.com under “Domain Name” so we can edit it. The Domain Name is the full domain name for a site, for example it might be djangobookstore.com, while the Display Name is a human-readable name for the site such as Django Bookstore.

Make these updates and click the “Save” button in the lower right corner when done.



Admin Sites - DjangoBookstore.com

Ok, back to our email. Let’s customize it a bit. On the first line we can see that this email actually extends another template–base\_message.txt–that contains the initial greeting of “Hello from…”. To update that we’d just need to add a base\_message.txt file to the email folder. Since this is just for demonstration purposes, trying changing “You’re” to “You are” to prove that we can customize the text.

email\_confirmation\_message.txt

{% extends "account/email/base\_message.txt" %}

{% load account %}

{% load i18n %}

{% block content %}{% autoescape off %}{% user\_display user as user\_display %}

{% blocktrans with site\_name=current\_site.name site\_domain=current\_site.domain %}

You are receiving this e-mail because user {{ user\_display }} has given your \

e-mail address to register an account on {{ site\_domain }}.

To confirm this is correct, go to {{ activate\_url }}\

{% endblocktrans %}{% endautoescape %}{% endblock %}

One final item to change. Did you notice the email was from webmaster@localhost? That’s a default setting we can also update via DEFAULT\_FROM\_EMAIL. Let’s do that now by adding the following line at the bottom of the django\_project/settings.py file.

# django\_project/settings.py

DEFAULT\_FROM\_EMAIL = "admin@djangobookstore.com" # new

Make sure you are logged out of the site and go to the Sign Up page again to create a new user. I’ve used testuser4@email.com for convenience.



Sign Up testuser4

Sign up and after being redirected to the homepage check the command line to see the message by typing docker-compose logs.

docker-compose logs

...

web\_1 | Content-Transfer-Encoding: 7bit

web\_1 | Subject: [Django Bookstore] Confirm Your Sign Up

web\_1 | From: admin@djangobookstore.com

web\_1 | To: testuser4@email.com

web\_1 | Date: Tue, 17 May 2022 18:34:50 -0000

web\_1 | Message-ID: <156312929025.27.2332096239397833769@87d045aff8f7>

web\_1 |

web\_1 | Hello from Django Bookstore!

web\_1 |

web\_1 | You are receiving this e-mail because user testuser4 has given your\

e-mail address to register an account on djangobookstore.com.

web\_1 |

web\_1 | To confirm this is correct, go to http://127.0.0.1:8000/accounts/\

Chapter 10: Email 154

confirm-email/Mg:1nr527:FhQTQdZha\_1mIsF9B5--71pfMDNlnR2vy4-sTrFmAyQ/

web\_1 |

web\_1 | Thank you from Django Bookstore!

web\_1 | djangobookstore.com

And there it is with the new From setting, the new domain djangobookstore.com, and the new message in the email.

**Email Confirmation Page**

Click on the unique URL link in the email which redirects to the email confirm page.



**Confirm Email Page**

**Note:** I don’t have the top “**Messages**” because nowadays it’s not showing but back then it does.

Not very attractive. Let’s update it to match the look of the rest of our site. Searching again in the django-allauth source code on Github reveals the name and location of this file is templates/account/email\_confirm.html. So let’s create our own template file with the same name and then update it to extend \_base.html and use Bootstrap for the button.

<!-- templates/account/email\_confirm.html -->

{% extends "\_base.html" %}

{% load i18n %}

{% load account %}

{% block head\_title %}{% trans "Confirm E-mail Address" %}{% endblock %}

{% block content %}

<h1>{% trans "Confirm E-mail Address" %}</h1>

{% if confirmation %}

{% user\_display confirmation.email\_address.user as user\_display %}

<p>{% blocktrans with confirmation.email\_address.email as email %}Please confirm

that <a href="mailto:{{ email }}">{{ email }}</a> is an e-mail address for user

{{ user\_display }}.{% endblocktrans %}</p>

<form method="post" action="{% url 'account\_confirm\_email' confirmation.key %}">

{% csrf\_token %}

<button class="btn btn-primary" type="submit">{% trans 'Confirm' %}</button>

</form>

{% else %}

{% url 'account\_email' as email\_url %}

<p>{% blocktrans %}This e-mail confirmation link expired or is invalid. Please

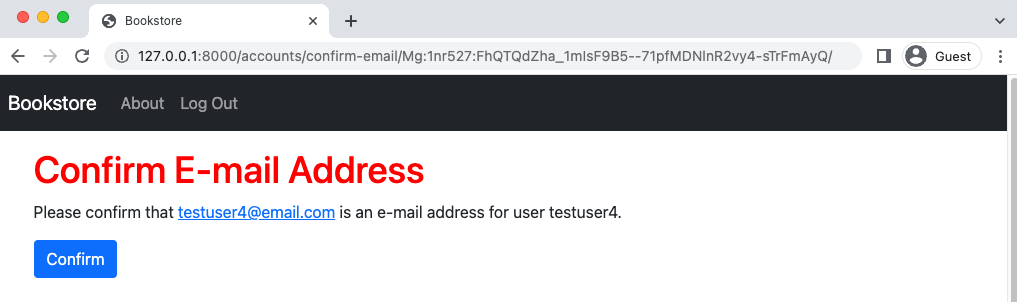
<a href="{{ email\_url }}">issue a new e-mail confirmation request</a>.\

{% endblocktrans %}</p>

{% endif %}

{% endblock %}

Refresh the page to see our update.



Confirm Email Page Updated

**Password Reset and Password Change**

Django and django-allauth also come with support for additional user account features such as the ability to reset a forgotten password and change your existing password if already logged in.

The locations of the default password reset and password change pages are as follows:

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

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

If you go through the flow of each you can find the corresponding templates and email messages in the django-allauth source code.

**Email Service**

The emails we have configured so far are generally referred to as “Transactional Emails” as they occur based on a user action of some kind. This is in contrast to “Marketing Emails” such as, say, a monthly newsletter.

There are many transactional email providers available including SendGrid, MailGun, Amazon’s Simple Email Service. Django is agnostic about which provider is used; the steps are similar for all and many have a free tier available.

After signing up for an account with your email service of choice you’ll often have a choice between using SMTP or a Web API. SMTP is easier to configure, but a web API is more configurable and robust. Start with SMTP and work your way from there: email configurations can be quite complex in their own right.

After obtaining a username and password with an email provider, a few settings tweaks will allow Django to use them to send emails.

The first step would be to update the EMAIL\_BACKEND config, which should be near the bottom of the django\_project/settings.py file since we previously updated it in the django-allauth config section.

# django\_project/settings.py

EMAIL\_BACKEND = "django.core.mail.backends.smtp.EmailBackend" # new

This means email will no longer output to the command line console but instead attempt to connect with an SMTP server. Then configure EMAIL\_HOST, EMAIL\_HOST\_USER, EMAIL\_HOST\_- PASSWORD, EMAIL\_PORT, and EMAIL\_USE\_TLS based on the instructions from your email provider

as environment variables.

In the official source code the EMAIL\_BACKEND will remain console, but the previous steps are how to add an email service. If you find yourself frustrated properly configuring email, well, you’re not alone! Django does at least make it far, far easier than implementing without the benefits of a batteries-included framework.

**Git**

To commit this chapter’s code updates make sure to check the status of changes, add them all, and include a commit message.

git init

git status

git add .

git commit -m “Chapter 10. Email”

If you have any issues compare your code against the official source code on Github.

**Conclusion**

Configuring email properly is largely a one-time pain. But it is a necessary part of any production website. This concludes the foundational chapters for our Bookstore project. In the next chapter we’ll finally start building out the Bookstore itself.