TRAINING DAY13 REPORT

07 JULY 2025

What is Email Sending in Django?

Django has a built-in email framework that allows developers to send emails directly from their web applications. This is often used for:

- Account verification links
- Password reset emails
- Notifications and updates
- Marketing or transactional messages

How It Works Internally

- Configuration – In settings.py, we define the email backend and server details (e.g., SMTP server for Gmail).
- Email Backend – Django's default SMTP backend uses Python's smtplib to connect to an email server.
- Authentication – Django logs into the email server using the username and password you provide.
- Sending Process – When you call send_mail() or similar functions, Django sends the email over a secure connection (TLS/SSL).
- Delivery – The email server (e.g., Gmail) handles final delivery to the recipient's inbox.

Email Backend in Django

Django supports multiple types of email backends for different purposes:

- SMTP Backend (django.core.mail.backends.smtp.EmailBackend) — Sends real emails via an SMTP server such as Gmail, Outlook, or a custom mail server.
- Console Backend (django.core.mail.backends.console.EmailBackend) — Prints email content to the console (useful for testing).
- File Backend (django.core.mail.backends.filebased.EmailBackend) — Stores emails as files instead of sending them.
- In-memory Backend (django.core.mail.backends.locmem.EmailBackend) — Keeps emails in memory for quick testing.

For production-level applications, the SMTP backend is most commonly used as it connects to a real mail server and delivers emails to actual recipients.

Gmail SMTP Configuration

To send emails through Gmail in Django, specific settings must be added to the settings.py file:

```
EMAIL_BACKEND = "django.core.mail.backends.smtp.EmailBackend"
EMAIL_HOST = "smtp.gmail.com"
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = "your_email@gmail.com"
EMAIL_HOST_PASSWORD = "your_app_password"
```

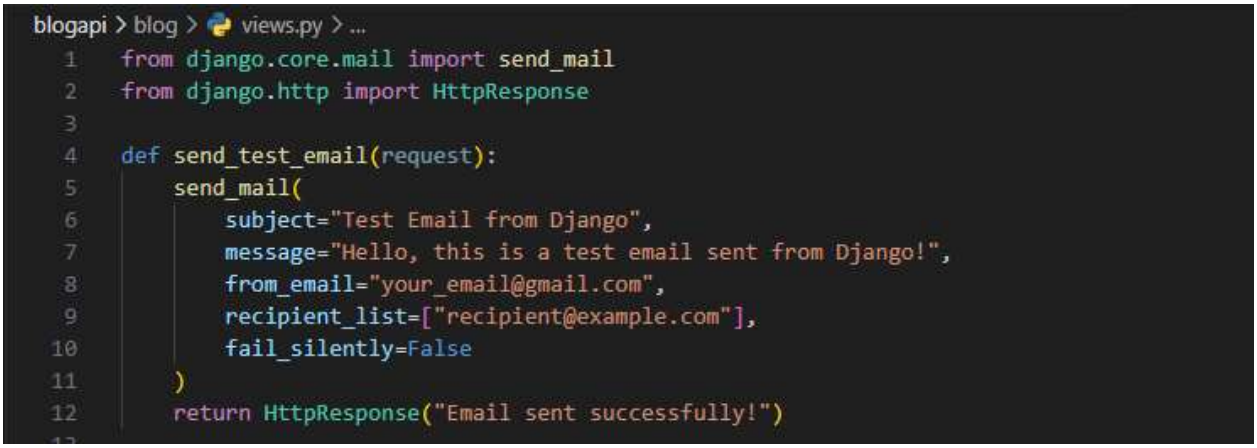
Explanation of Settings:

- **EMAIL_BACKEND** → Specifies the backend Django will use for sending emails. Here, it's configured for SMTP.
- **EMAIL_HOST** → The address of the mail server (for Gmail, it is smtp.gmail.com).
- **EMAIL_PORT** → The port number for the connection. Port 587 is used for TLS encryption.
- **EMAIL_USE_TLS** → Enables Transport Layer Security for secure transmission.
- **EMAIL_HOST_USER** → Your Gmail account from which emails will be sent.
- **EMAIL_HOST_PASSWORD** → App password generated from your Google account for authentication.

Sending Emails

1. **Sending a Plain Text Email**

Example using Django's built-in send_mail() function:



This function takes the subject, message, sender, recipient list, and a flag to decide whether errors should raise exceptions.

2. **Sending an HTML Email**

To send rich content with HTML formatting:

```
blogapi > blog > views.py > ...
1  from django.core.mail import EmailMultiAlternatives
2
3  subject = "HTML Email from Django"
4  text_content = "This is the plain text version."
5  html_content = "<h1>Hello!</h1><p>This is an <b>HTML</b> email from Django.</p>"
6
7  msg = EmailMultiAlternatives(subject, text_content, "your_email@gmail.com", ["recipient@example.com"])
8  msg.attach_alternative(html_content, "text/html")
9  msg.send()
10
```

In urls.py

Link the view to a URL so you can test it:

```
blogapi > blog > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path("send-email/", views.send_test_email, name="send_email"),
6  ]
7
```

Security Note for Gmail Users

Google no longer allows “Less Secure Apps” to access accounts. To use Gmail SMTP:

1. Enable Two-Step Verification in your Google account.
2. Generate an App Password from Google account security settings.
3. Use the App Password in EMAIL_HOST_PASSWORD instead of your regular password.

This ensures a secure authentication process and prevents unauthorized access.

FLOWCHART DESCRIBING STEPS NEEDED FOR SENDING EMAILS