Django Email Configuration

# 1. Email Configuration in settings.py

In Django, email settings are usually defined inside the settings.py file. Below are the commonly used configurations:

**EMAIL\_BACKEND:** Specifies the backend Django will use to send emails. Commonly set to 'django.core.mail.backends.smtp.EmailBackend'. For testing, you can use 'console.EmailBackend'.

**EMAIL\_HOST:** The hostname of your email server (e.g., 'smtp.gmail.com' for Gmail, 'smtp.office365.com' for Outlook).

**EMAIL\_PORT:** The port used for sending emails. Common ports are 587 (TLS), 465 (SSL), and 25 (legacy, often blocked).

**EMAIL\_USE\_TLS:** Set to True to use TLS (Transport Layer Security) encryption. Usually used with port 587.

**EMAIL\_USE\_SSL:** Set to True to use SSL (Secure Sockets Layer). Usually used with port 465. Do not enable both TLS and SSL together.

**EMAIL\_HOST\_USER:** The email address used to send emails (e.g., your Gmail address).

**EMAIL\_HOST\_PASSWORD:** The password or App Password for your email account. For Gmail, you must enable 2FA and use an App Password.

**DEFAULT\_FROM\_EMAIL:** The default 'from' address used when sending emails if not explicitly provided.

# 2. Sending Emails in Django

Once configuration is set, you can send emails using Django’s built-in utilities.

## 2.1 Sending a Simple Email

The simplest way is to use send\_mail() from django.core.mail:

from django.core.mail import send\_mail  
from django.conf import settings  
  
send\_mail(  
 subject="Welcome",  
 message="This is a test email from Django.",  
 from\_email=settings.DEFAULT\_FROM\_EMAIL,  
 recipient\_list=["recipient@example.com"]  
)

## 2.2Sending Email with Attachment

You can attach files using EmailMessage:

from django.core.mail import EmailMessage  
from django.conf import settings  
  
email = EmailMessage(  
 subject="Report",  
 body="Here is your report.",  
 from\_email=settings.DEFAULT\_FROM\_EMAIL,  
 to=["recipient@example.com"]  
)  
email.attach\_file("path/to/file.pdf")  
email.send()

## 2.3 Development and Testing Options

During development, you may not want to actually send emails. Django provides testing backends:

- Console backend: prints email content in terminal (EMAIL\_BACKEND = 'django.core.mail.backends.console.EmailBackend').  
- File backend: writes emails to files for inspection.  
- Locmem backend: stores emails in memory for use in unit tests.

# 3. Why Port 587?

Port 587 is the modern standard for email submission with STARTTLS. It is recommended by IETF and is widely supported. Port 465 is used for SSL connections but considered legacy. Port 25 is often blocked by ISPs and mainly used for server-to-server communication.

# 4. Best Practices

- Always use App Passwords or API keys instead of your main email password.  
- Use environment variables to store secrets.  
- Include both plain-text and HTML versions for better deliverability.  
- Use async processing (e.g., Celery) for sending large volumes of emails.  
- Configure SPF, DKIM, and DMARC if sending from a custom domain.

# 5. Example Configuration Code

Below is a sample configuration snippet to add in your settings.py file:

# settings.py  
  
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" # Use App Password, not your normal password  
DEFAULT\_FROM\_EMAIL = EMAIL\_HOST\_USER