

# Managing Environment Variables

# Step-by-Step Guide

## 1. Create a Base Settings File

First, create a `base.py` file where you'll put all the common settings for your Django project.

```
# settings/base.py
import os
from decouple import config

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

SECRET_KEY = config('SECRET_KEY')

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': config('DB_NAME'),
        'USER': config('DB_USER'),
        'PASSWORD': config('DB_PASSWORD'),
        'HOST': config('DB_HOST'),
        'PORT': config('DB_PORT'),
    }
}
```

## 2. Create a `.env` File

Next, create a `.env` file at the root of your project directory to store sensitive credentials and environment-specific settings.

```
# .env
SECRET_KEY=your-secret-key
DB_NAME=your-database-name
DB_USER=your-database-user
DB_PASSWORD=your-database-password
DB_HOST=your-database-host
DB_PORT=your-database-port
```

### 3. Install Python-Decouple

Install `python-decouple` to help manage environment variables.

```
pip install python-decouple
```

### 4. Create Environment-Specific Settings Files

Create a `settings` directory and add environment-specific settings files `development.py`, `production.py`, and `testing.py`. These files will extend `base.py` and override settings as needed.

## Directory Structure

```
myproject/  
  settings/  
    __init__.py  
    base.py  
    development.py  
    production.py  
    testing.py  
  .env  
  manage.py
```

## Example development.py

```
# settings/development.py
from .base import *

DEBUG = True

ALLOWED_HOSTS = []

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

## Example `production.py`

```
# settings/production.py
from .base import *

DEBUG = False

ALLOWED_HOSTS = ['your-production-domain.com']

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': config('DB_NAME'),
        'USER': config('DB_USER'),
        'PASSWORD': config('DB_PASSWORD'),
        'HOST': config('DB_HOST'),
        'PORT': config('DB_PORT'),
    }
}
```



## Example `testing.py`

```
# settings/testing.py
from .base import *

DEBUG = False

ALLOWED_HOSTS = []

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'test_db.sqlite3'),
    }
}
```

## 5. Setting the Environment Variable

You need to tell Django which settings file to use by setting the `DJANGO_SETTINGS_MODULE` environment variable.

### Command Line

```
# For development
export DJANGO_SETTINGS_MODULE=myproject.settings.development

# For production
export DJANGO_SETTINGS_MODULE=myproject.settings.production

# For testing
export DJANGO_SETTINGS_MODULE=myproject.settings.testing

# Run Django server
python manage.py runserver
```

## Virtual Environment Activation Script

For Unix-like systems

Edit the `bin/activate` file in your virtual environment directory

```
# Add this line to bin/activate  
export DJANGO_SETTINGS_MODULE=myproject.settings.development
```

For Windows

Edit the `Scripts/activate.bat` file

```
# Add this line to Scripts/activate.bat  
set DJANGO_SETTINGS_MODULE=myproject.settings.development
```

## Using a `.env` File

Load the `DJANGO_SETTINGS_MODULE` from your `.env` file

```
# .env
DJANGO_SETTINGS_MODULE=myproject.settings.development
```

And modify `base.py` to read the environment variable

```
# settings/base.py
import os
from decouple import config

os.environ.setdefault('DJANGO_SETTINGS_MODULE', config('DJANGO_SETTINGS_MODULE', default='myproject.settings.development'))
```

## Summary

1. Create a `base.py` for common settings.
2. Create a `.env` file for sensitive credentials.
3. Install `python-decouple` and use it to load environment variables in `base.py`.
4. Create environment-specific settings files ( `development.py` , `production.py` , `testing.py` ) that extend `base.py`.
5. Set the `DJANGO_SETTINGS_MODULE` environment variable in your terminal, virtual environment activation script, or `.env` file to switch between environments.

This approach helps you maintain clean and organized settings, easily switch between different environments, and keep sensitive information secure.