https://www.halvorsen.blog

Django

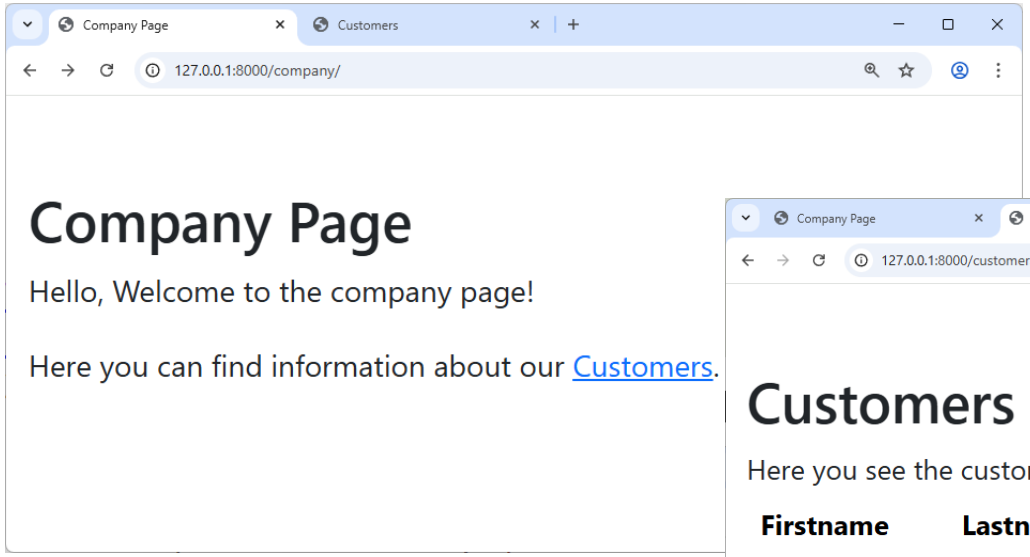
Python Web Development Framework

Hans-Petter Halvorsen

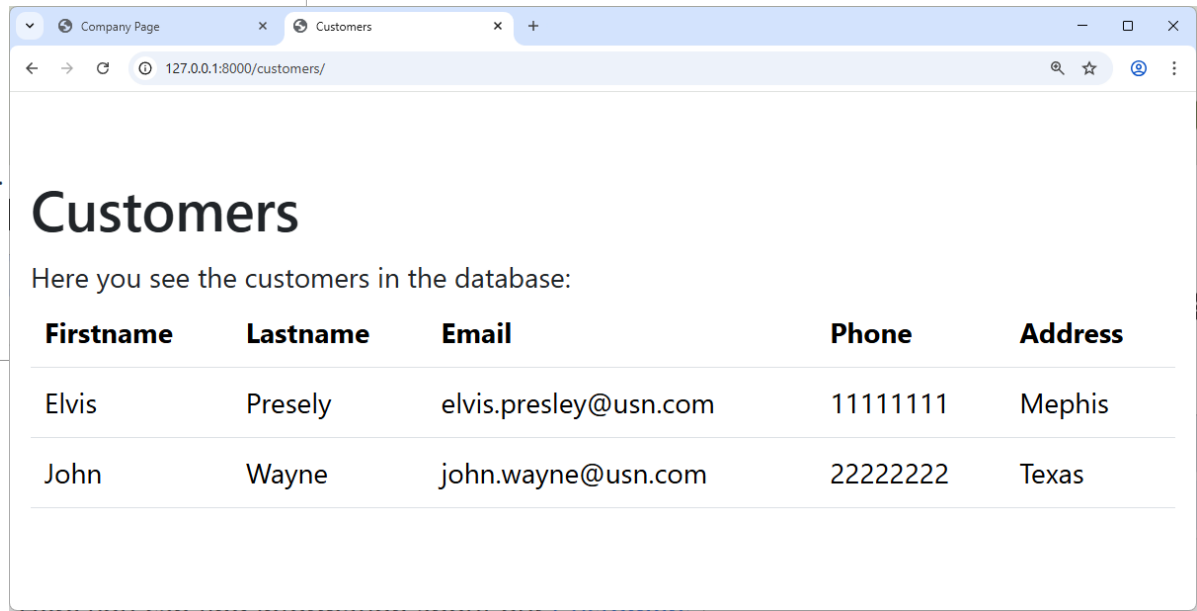
Contents

Django “Company” App

Main “Company” App:



Subpage showing the “Customers” for the “Company” App:



Introduction

Hans-Petter Halvorsen

Table of Contents

Introduction

- There are many web development frameworks out there.
- Examples are PHP, ASP.NET Core, Django, etc.
- The focus in this tutorial is Django.
- Django is a server-side web development framework where you use **Python** as programming language.

Python Web Frameworks

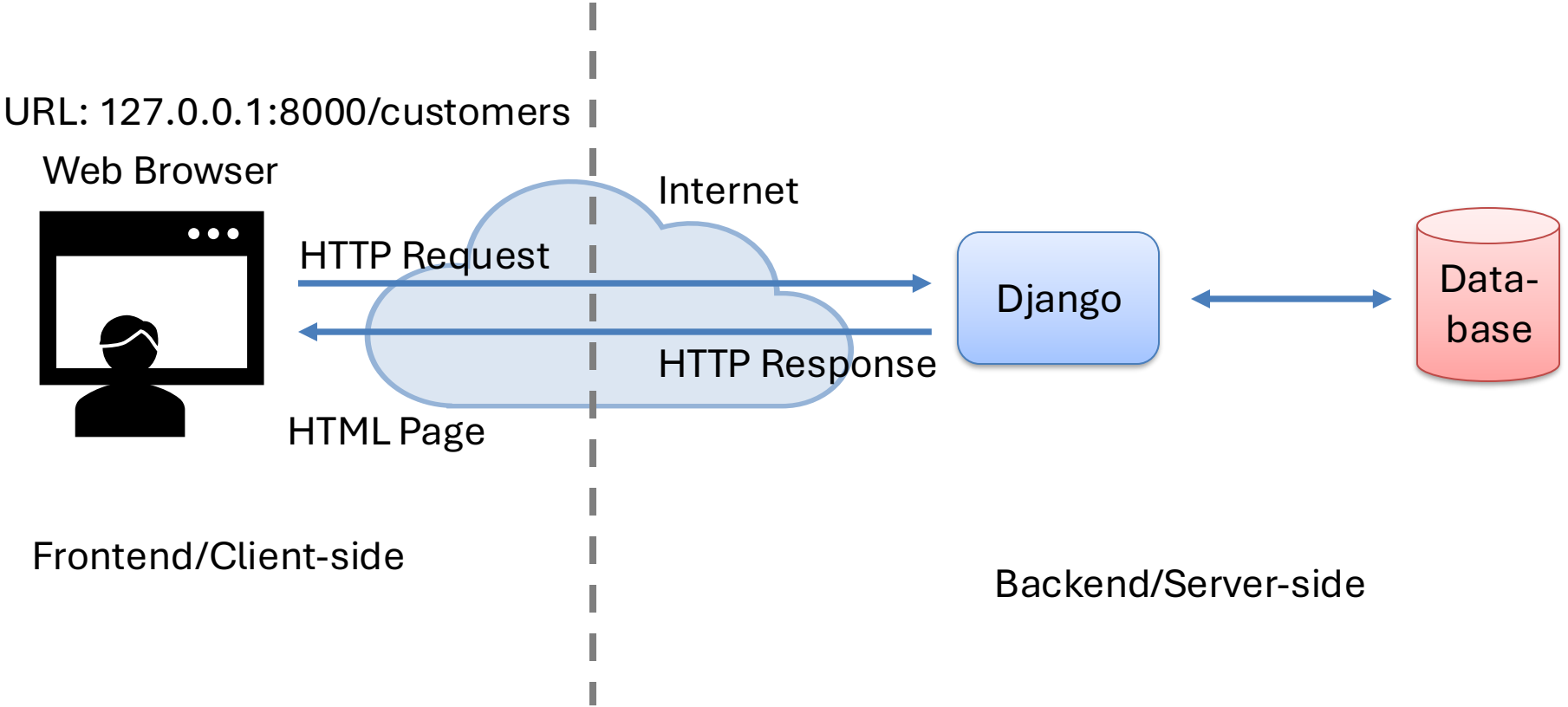
- **Django**
  - Probably the most popular Python Web Framework
  - Django will be the focus in this tutorial
- **Flask**
- Pyramid
- TurboGears
- Bottle
- +++

https://djangostars.com/blog/python-web-development/ and https://www.browserstack.com/guide/top-python-web-development-frameworks

Django

- Django is a Python web development framework.
- Django is a back-end/server-side web framework.
- Django has support for databases like SQLite, MySQL/MariaDB and PostgreSQL.
- Django includes an SQLite database, but for the others you need to install the database system you want to use from their respective websites and in addition install a Python package/driver for the specific database system.
- Django is free, open source and written in Python.
- Homepage: https://www.djangoproject.com

Django Workflow



Code Editors for Django

You can use any kind of tool/IDE for Python(Django development.
Here are some recommendations:

- **Visual Studio Code**.
  - Multiplatform and Free.
  - Homepage: https://code.visualstudio.com
- PyCharm IDE.
  - Free Community Edition
  - Free access to all JetBrains IDEs for students and educators.
  - Homepage: https://www.djangoproject.com
- Visual Studio (it has a Django Template). Free Community Edition.
- PyDev is another alternative.
  - https://www.pydev.org
- +++

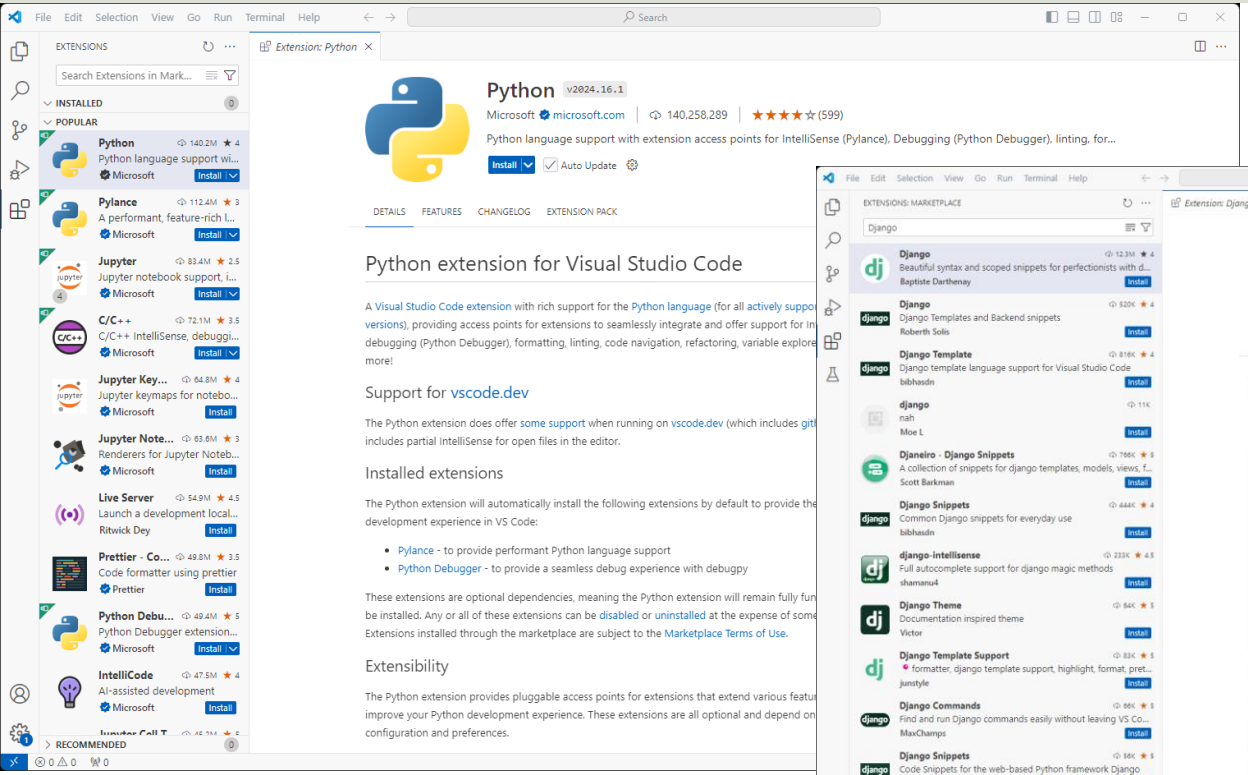
In this Tutorial we will focus on Visual Studio Code and setup the Django Environment using the Command Prompt.

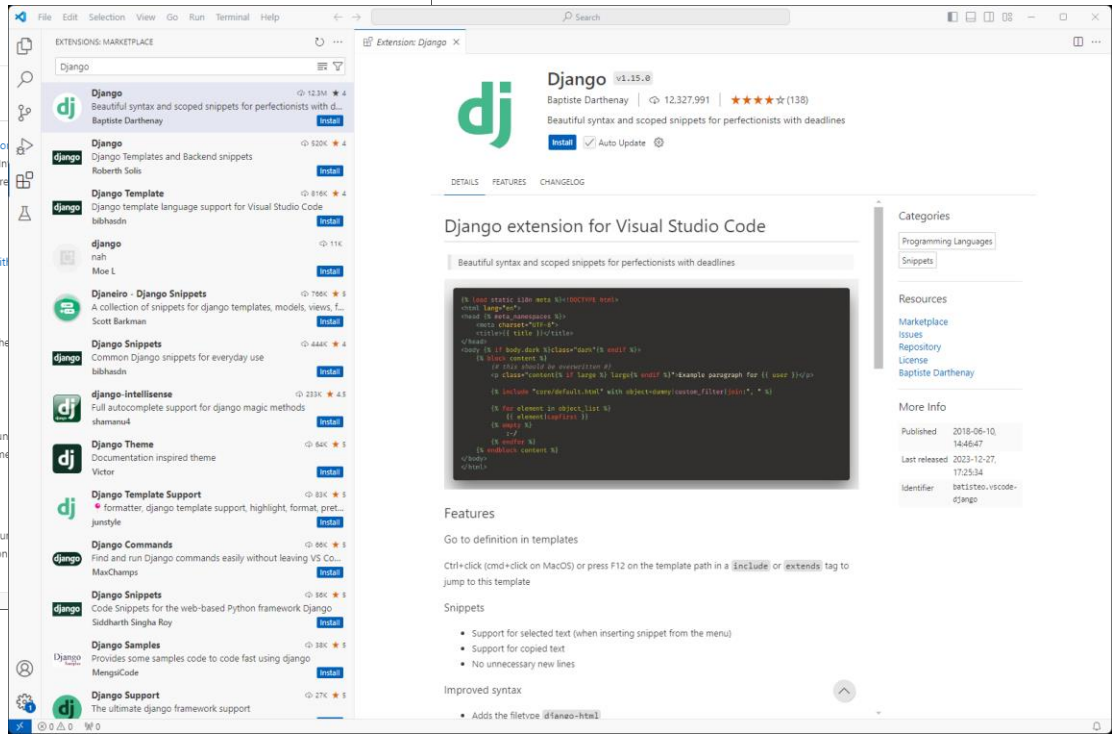
Note! Visual Studio is a “Code Editor”, while PyCharm and Visual Studio is an “IDE”.

Visual Studio Code

- Visual Studio Code is a free and open-source code editor developed by Microsoft.
- Cross-platform, works for Windows, Linux and macOS.
- It supports many programming languages, including Python, through an extension.
- Visual Studio Code is a general-purpose Code Editor for almost any kind of programming language or framework through so-called extensions that you can install on top of the IDE.
- The “Python” extension from Microsoft is needed for general Python development in Visual Studio Code.
- In addition, there are dozens of Extensions for Django development as well.

Visual Studio Code Extensions

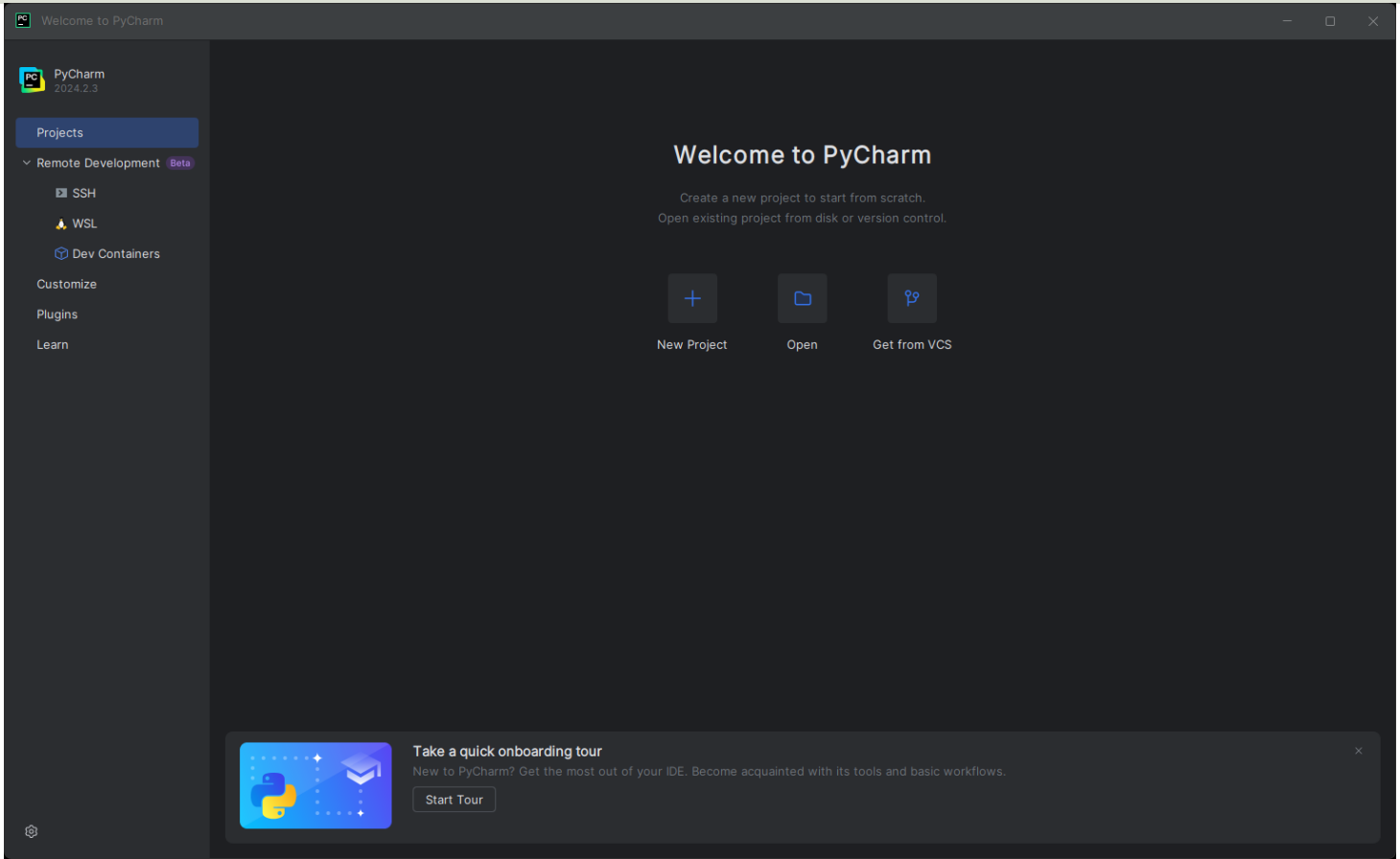




PyCharm

- PyCharm is a Python IDE developed by JetBrains.
- PyCharm is tailormade for Python development (as the name indicates).
- Cross-platform, works for Windows, Linux and macOS.
- It is one of the most popular Python IDEs.
- Free and paid version, Free Community Edition and free Pro version for students and educators.
- PyCharm also supports web development with Python, offering compatibility with frameworks like Django and Flask.
- Homepage: https://www.djangoproject.com

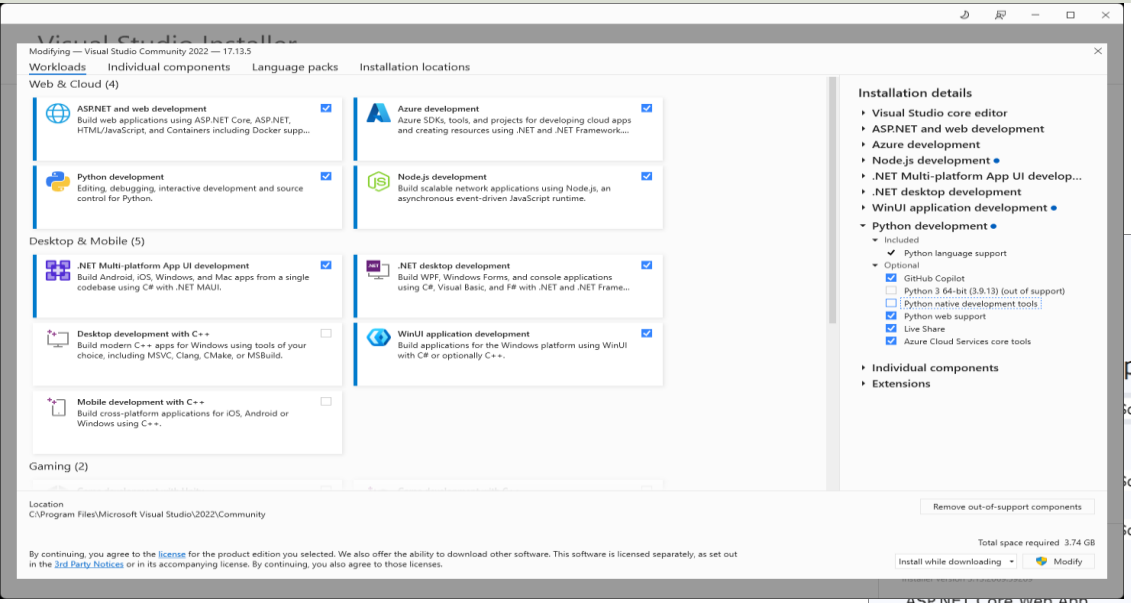
PyCharm

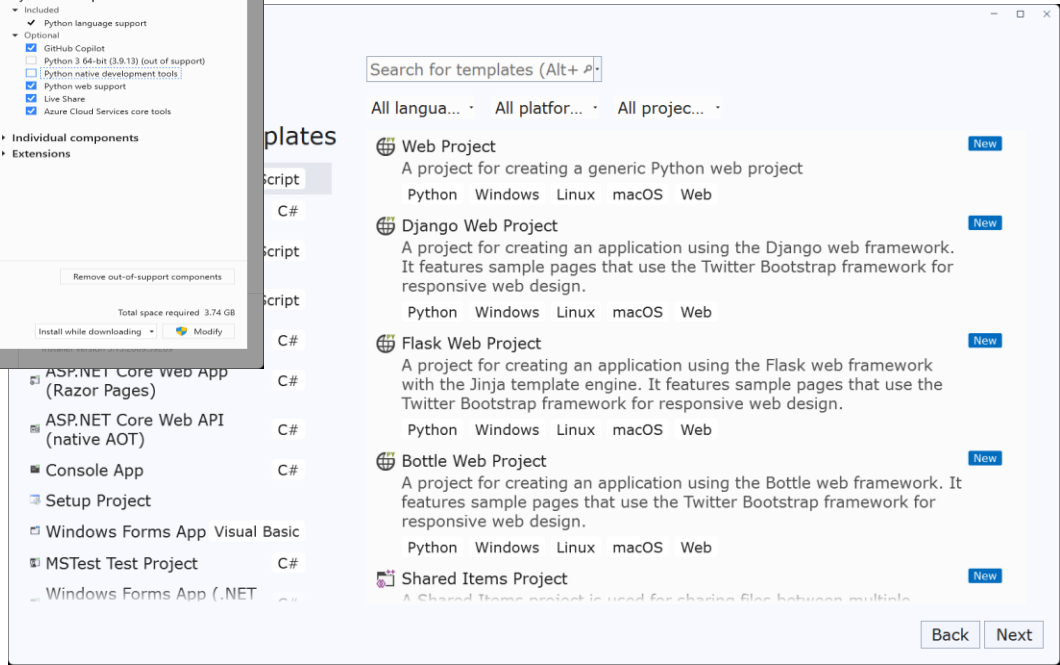

Welcome to PyCharm
PyCharm
2024.2.3
Projects
Remote Development
Beta
SSH
WSL
Dev Containers
Customize
Plugins
Learn
Welcome to PyCharm
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
Take a quick onboarding tour
New to PyCharm? Get the most out of your IDE. Become acquainted with its tools and basic workflows.
Start Tour

Visual Studio

- Visual Studio is an Integrated Development Environment (IDE), while Visual Studio Code is a Code Editor.
- Using a fullblood IDE compared to a Code Editor like Visual Studio Code makes it easier to setup a new Django Project because you can do more things from a GUI instead of using the Command Prompt/Terminal (CLI, Command Line Interface).
- The Django Template in Visual Studio comes also with additional features including built-in support for Bootstrap, etc.

Visual Studio





Django Resources

- **Django Homepage**: https://www.djangoproject.com
- **Getting started with Django**: https://www.djangoproject.com/start/
- **Django Tutorial w3schools**: https://www.w3schools.com/django/
- Choosing Between PyCharm and VS Code: https://blog.neverinstall.com/the-best-python-ide-from-your-browser/
- Setting up VSCode for Python: A Complete Guide: https://www.datacamp.com/tutorial/setting-up-vscode-python?ref=blog.neverinstall.com
- Python for Web Development: Pros and Cons: https://djangostars.com/blog/python-web-development/

https://www.halvorsen.blog

Getting Started with Django

Hans-Petter Halvorsen

Table of Contents

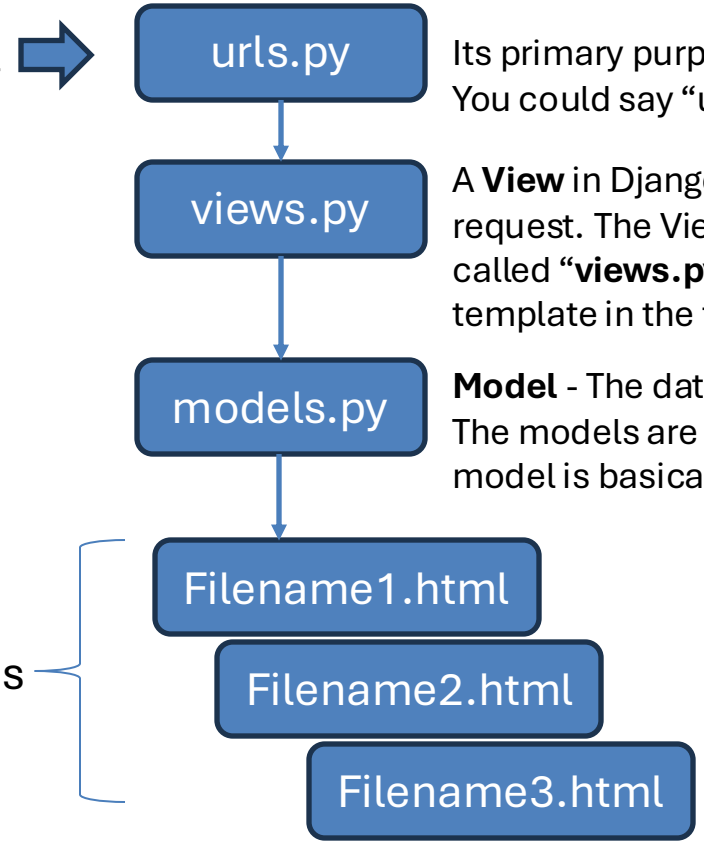
Model View Template (MVT)

Django follows the MVT design pattern (Model View Template).

- **Model** - The data you want to present, usually data from a database. The models are usually located in a file called “models.py”.
- **View** - A request handler that returns the relevant template and content - based on the request from the user. The views are usually located in a file called “views.py”.
- **Template** – A HTML file containing the layout of the web page, with logic on how to display the data. The HTML template files are stored in a subfolder called “**templates**”

Django Files and Folders

URL entered in the Web Browser



Its primary purpose is to map URL patterns to their corresponding views. You could say “urls.py” is the table of contents for the Django project.

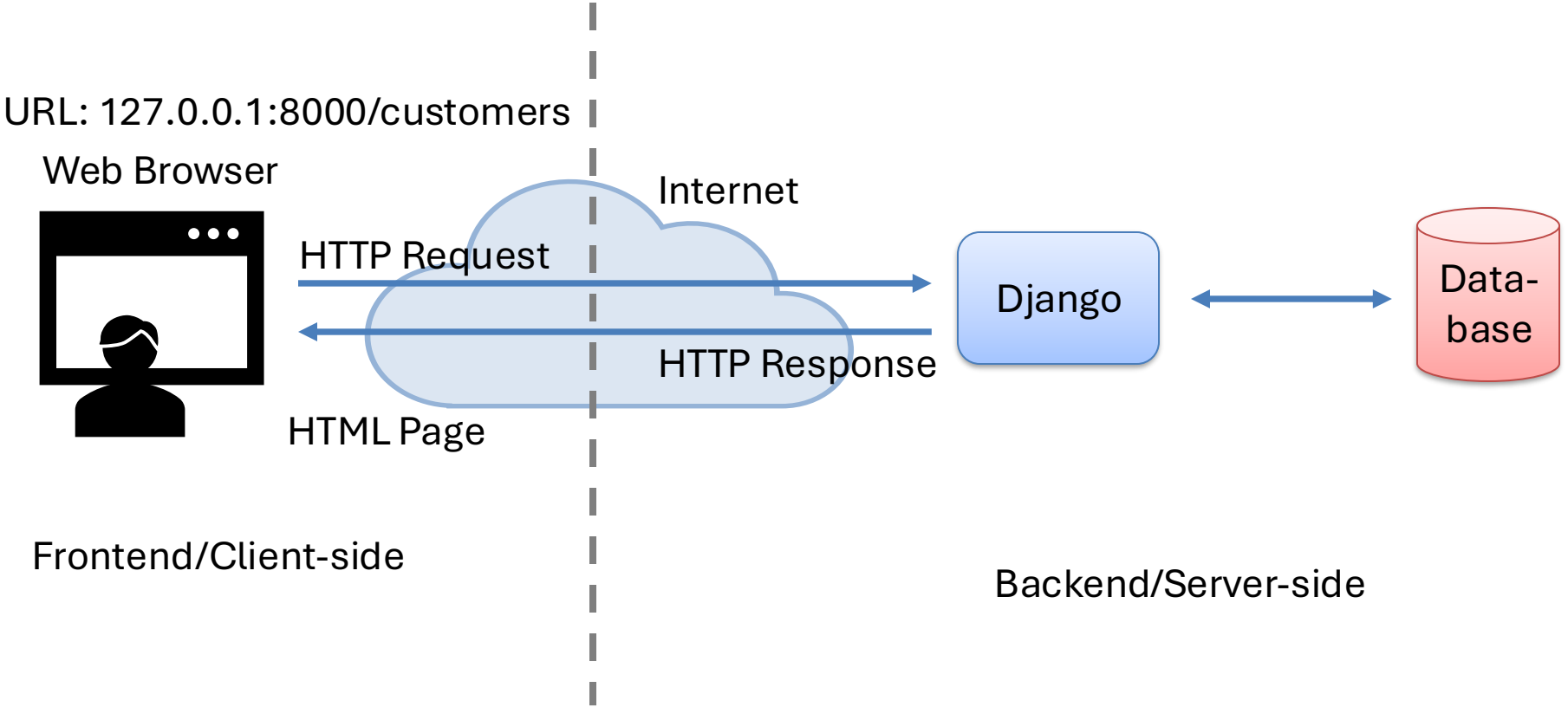
A **View** in Django is basically a Python function that handles an HTTP request. The Views (Python Functions) are usually located in a file called “**views.py**”. The view then sends the data to a specified template in the template folder.

Model - The data you want to present, usually data from a database. The models are usually located in a file called “**models.py**”. A Django model is basically a table in your database.

Template Files

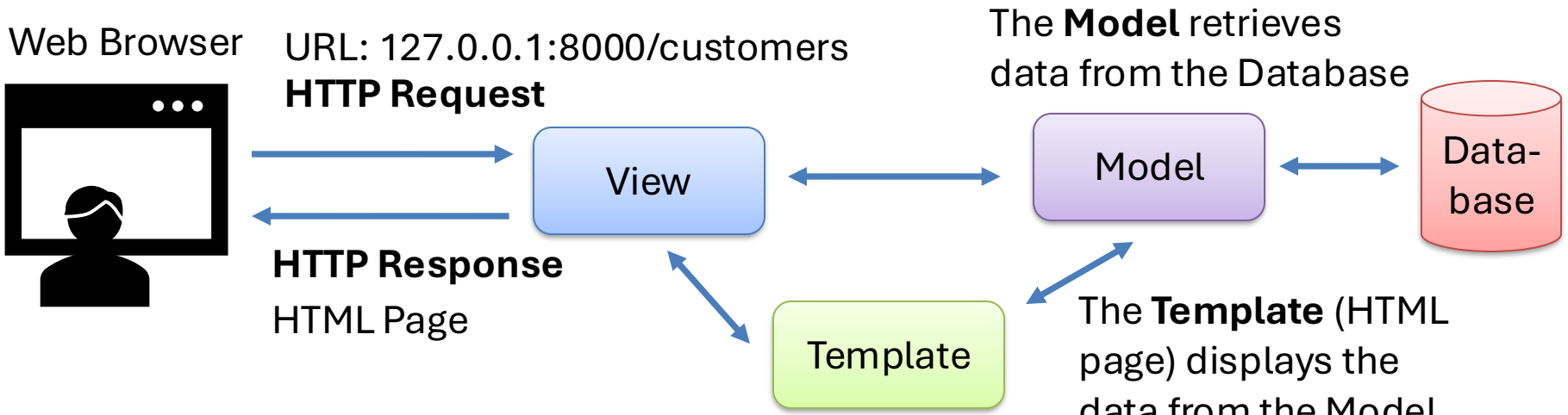
Django Templates are HTML files containing the layout of the web pages, with logic on how to display the data
The HTML template files are stored in a subfolder called “**templates**”

Django Workflow



Django Workflow

Django find which View to call based on information in “urls.py”.



The **View** handles the HTTP request from the Web Browser and sends data to the specific Template. Then the HTTP response is sent back to the Web Browser.

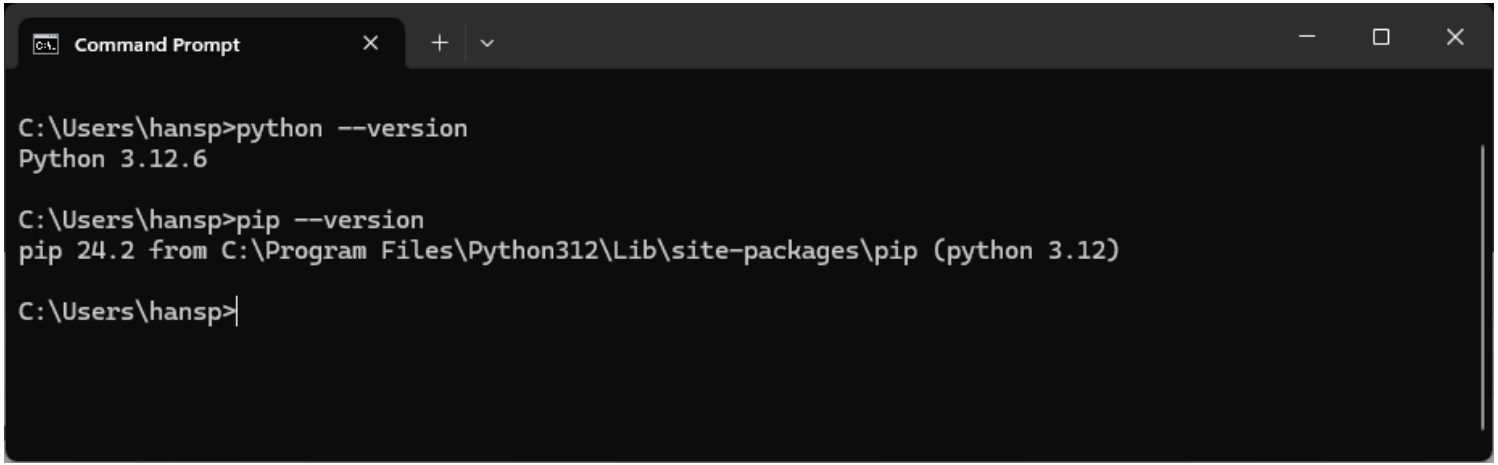
Python Files and Folders

- manage.py – This is used to run administrative commands, like running the Django project, create a Django App, etc.
- settings.py – Here you define different settings for the Django Project.
- urls.py - Its primary purpose is to map URL patterns to their corresponding views.
- views.py - A View in Django is basically a Python function that handles an HTTP request. The Views (Python Functions) are usually located in a file called “views.py”. The view then sends the data to a specified template in the template folder.
- models.py - Model - The data you want to present, usually data from a database. A Django model is basically a table in your database. The models are usually located in a file called “models.py”.
- /templates/ – A subfolder with all the HTML Template Files. Template files are HTML files containing the layout of the different web pages, with logic on how to display the data.

Python and Command Prompt

Note! Before you start installing Django you need to have **Python** installed.

Check if you have **Python** and **PIP** installed using the **Command Prompt**:

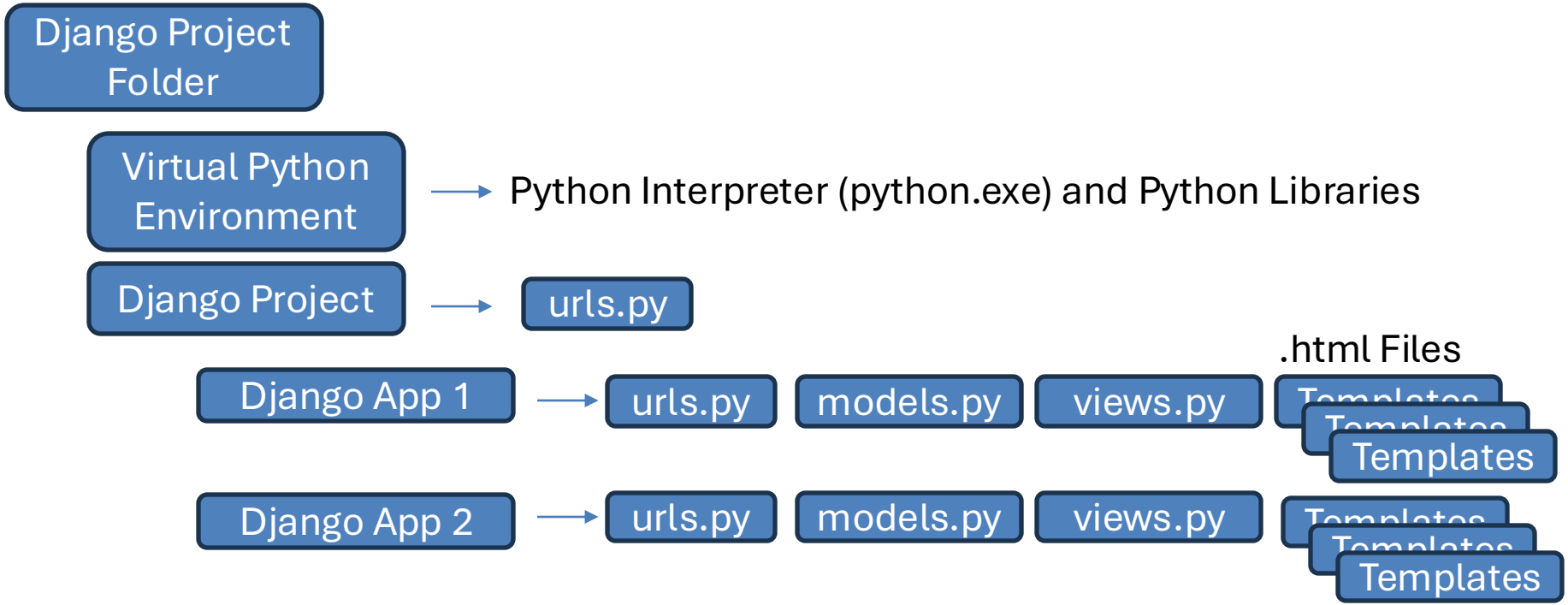


Typically, you want to create a dedicated **virtual Python environment** for each Django project.

You can also use the **Terminal**.

Django Projects and Django Apps

You start by creating a Django Project. Then you can create one or more Django Apps inside that Django Project.



https://www.halvorsen.blog

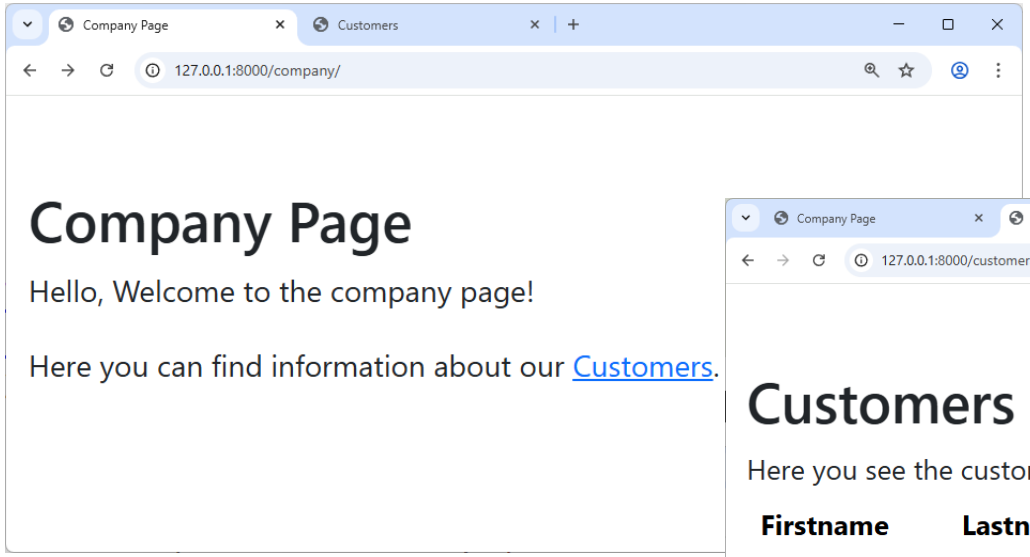
Create Django Project and Django App

Table of Contents

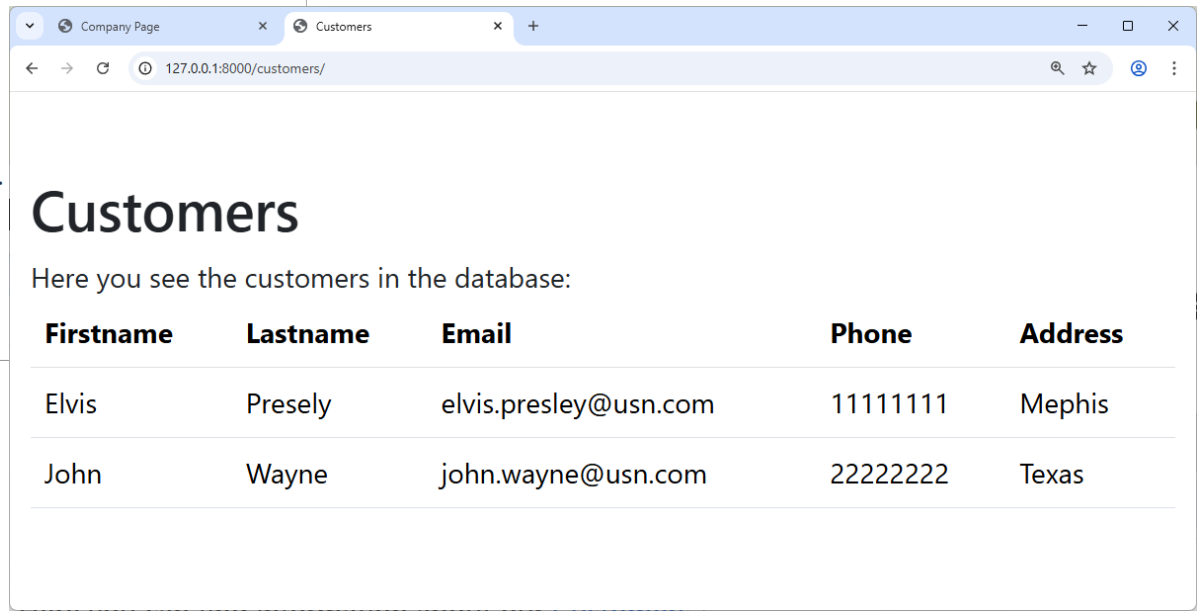
Hans-Petter Halvorsen

Django “Company” App

Main “Company” App:



Subpage showing the “Customers” for the “Company” App:



Introduction

We will create a basic **Django App** called “company”.

1. First, we need to create a **Virtual Python Environment**.
2. Second, we need to create a **Django Project**.
3. Third, then we can create the **Django App** inside the Django Project.
   - In a Django Project we can create multiple Django Apps.

Create Django Project and App

1. Create and Activate Virtual Python Environment

```
>python –m venv venvname
```

```
C:\...\venvname\Scripts>activate.bat
```

2. Install Django

```
...>python -m pip install Django
```

3. Create Django Project

```
...>django-admin startproject projectname
```

4. Run the Django Project

```
...>python manage.py runserver
```

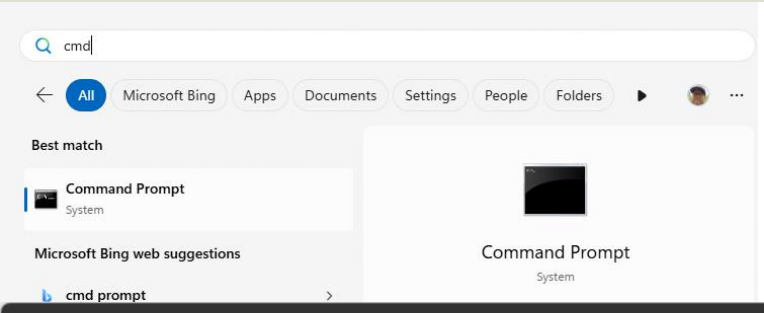
5. Create a Django App

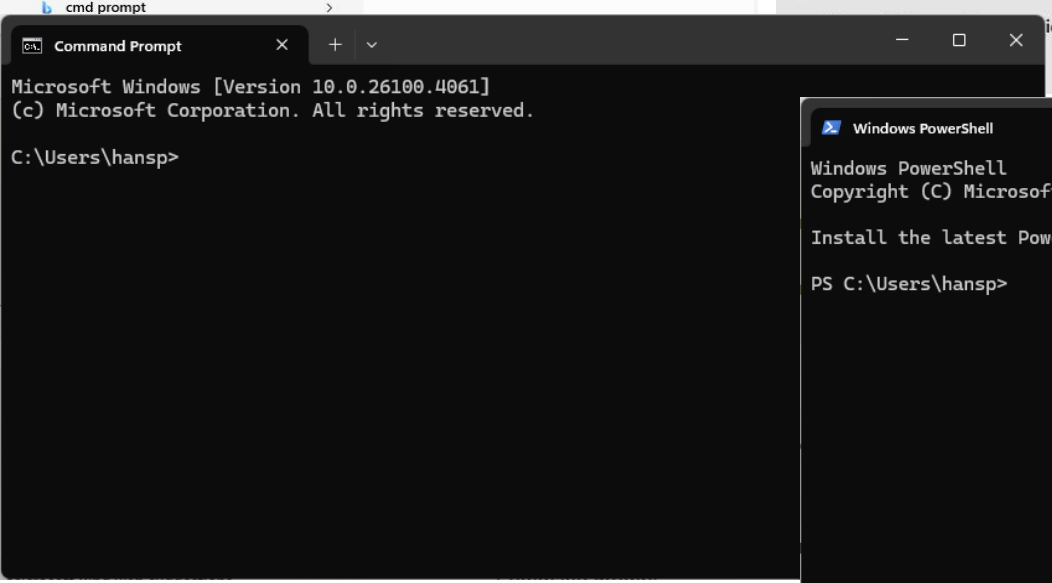
```
...>python manage.py startapp appname
```

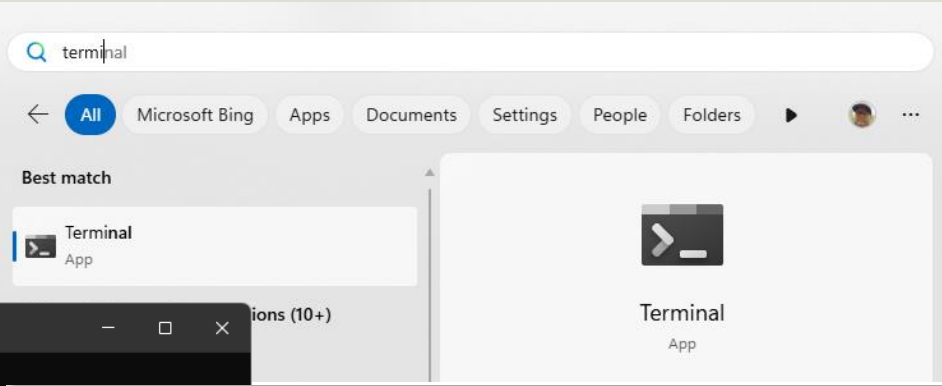
We need to do these steps to create a virtual Python environment (which is recommended), then install Django. Then you need to create a new Django Project and a Django App.

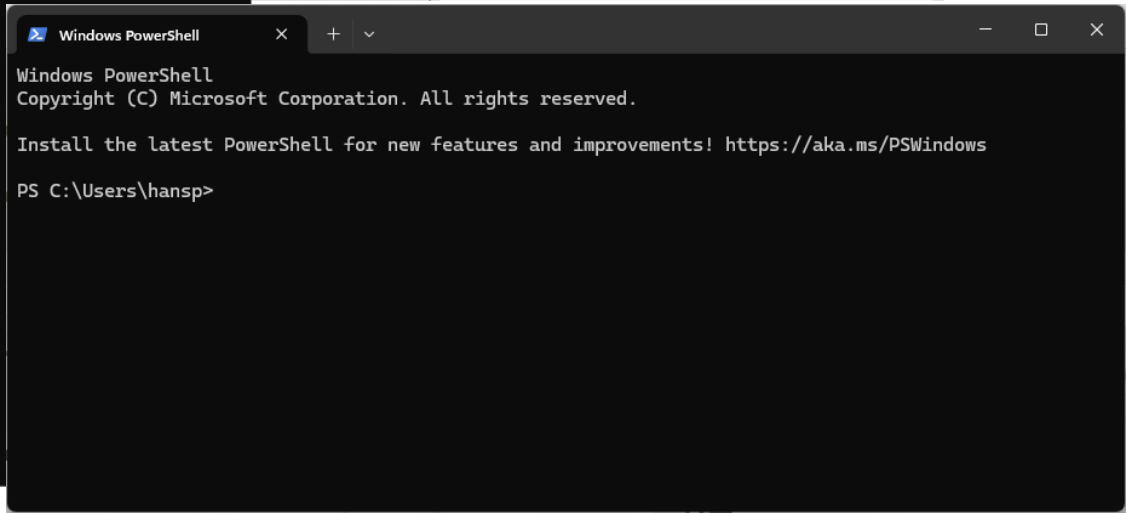
We can use the Command Prompt or the Terminal in Windows. We can also use the built-in Terminal in Visual Studio Code.

Command Prompt and Terminal





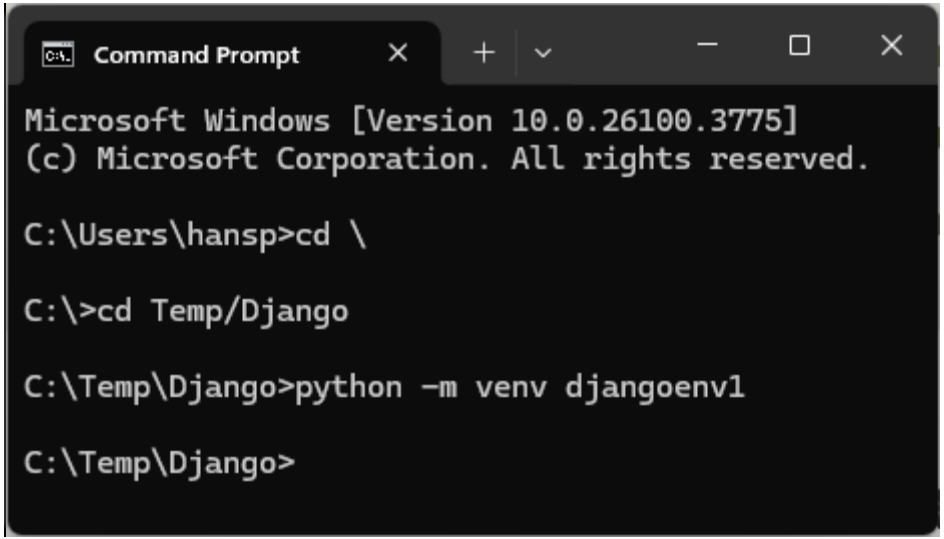




Create **Virtual** Python Environment

We start by creating a Folder where we want to create the Virtual Python Environment and the Django Application

In my example, I have created a folder C:/Temp/Django/

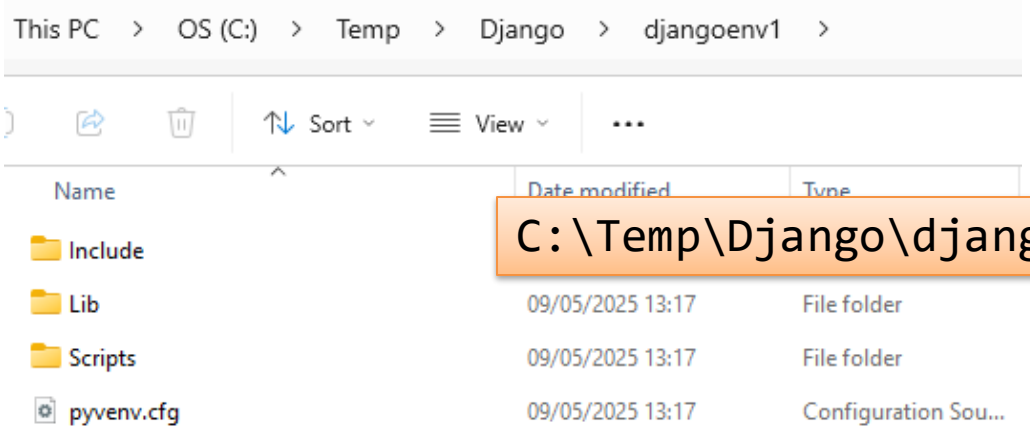



```
>python -m venv djangoenv1
```

This command will create a Virtual Python Environment called “djangoenv1”

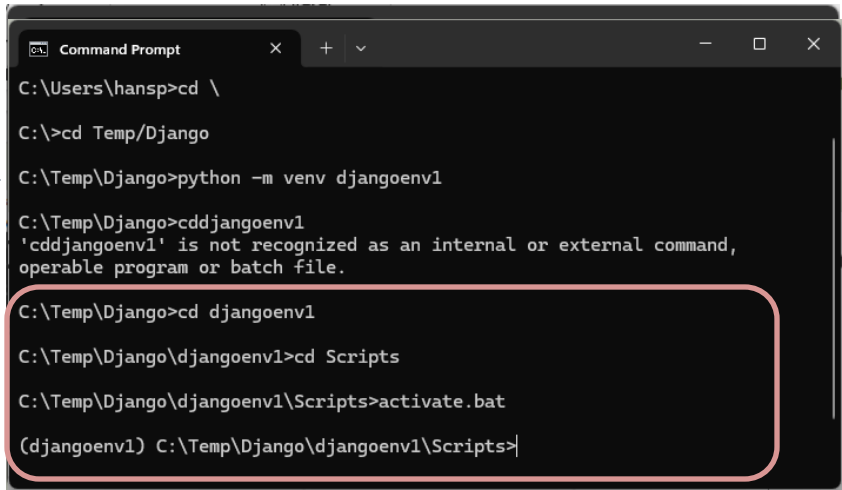
Activate Virtual Python Environment

Here we see the files created as part of the Virtual Python Environment:




```
C:\Temp\Django\djangoenv1\Scripts>activate.bat
```

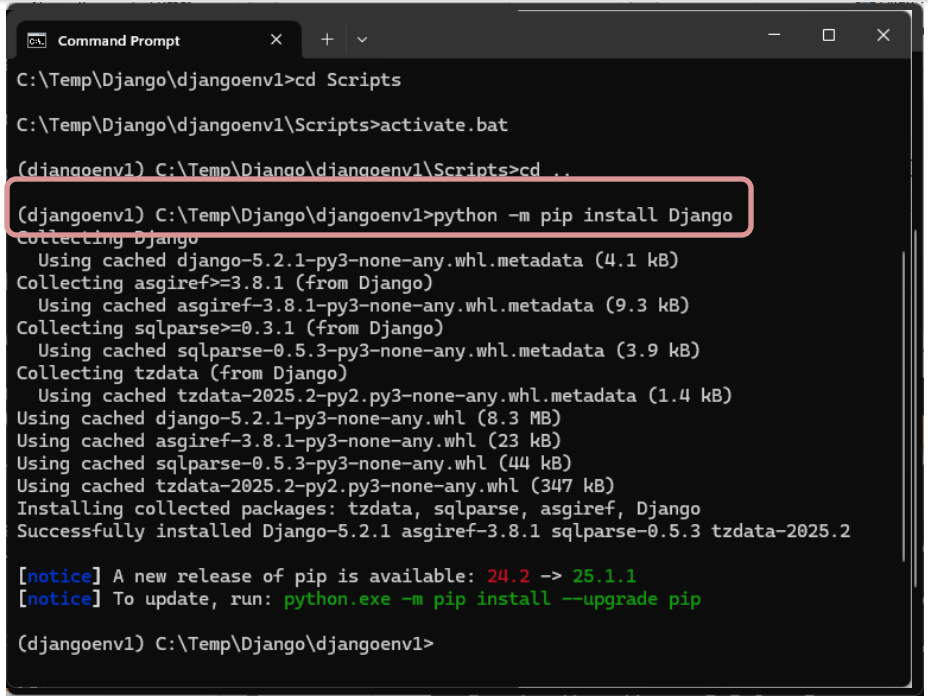
Next, we need to **activate** the Virtual Python Environment by running the "**activate.bat**" file located in the /Scripts folder:

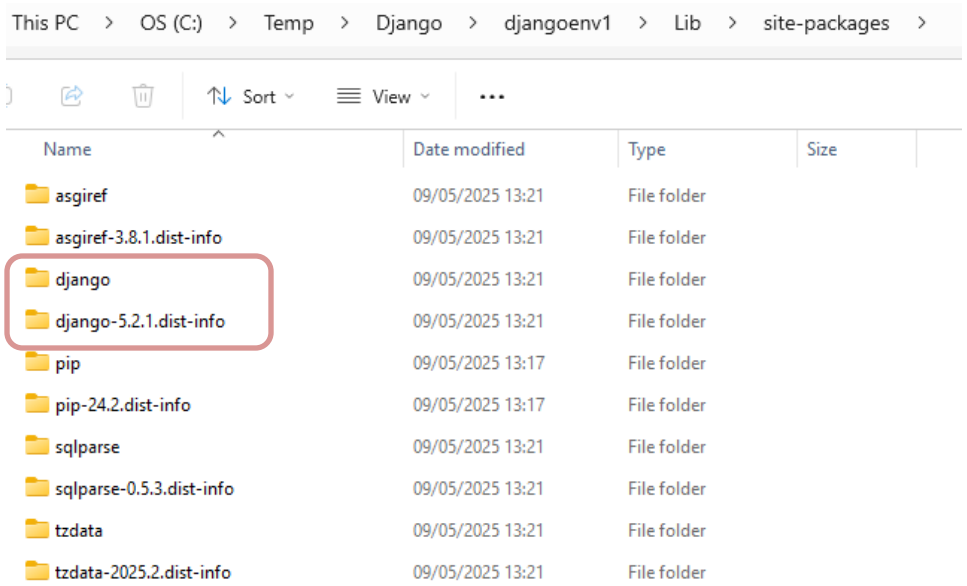


Note! You need to activate it every time you shall work on your project from the command prompt.

Install Django

```
(djangoenv1) C:\Temp\Django\djangoenv1>python -m pip install Django
```

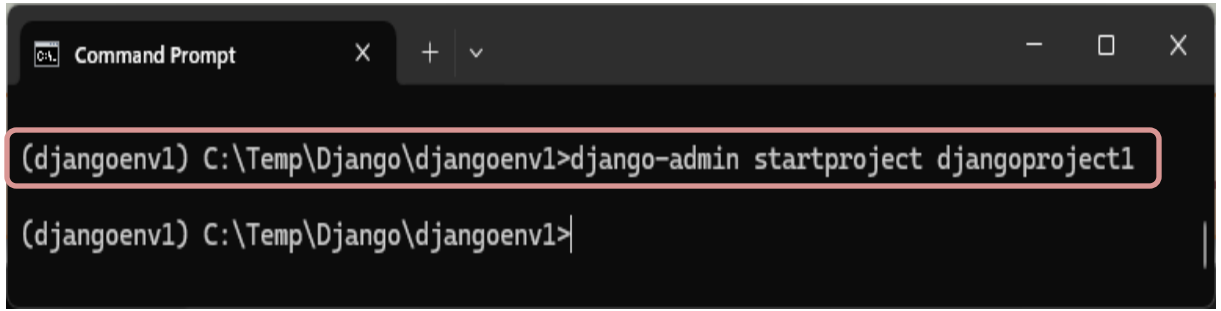




We see that Django is installed:

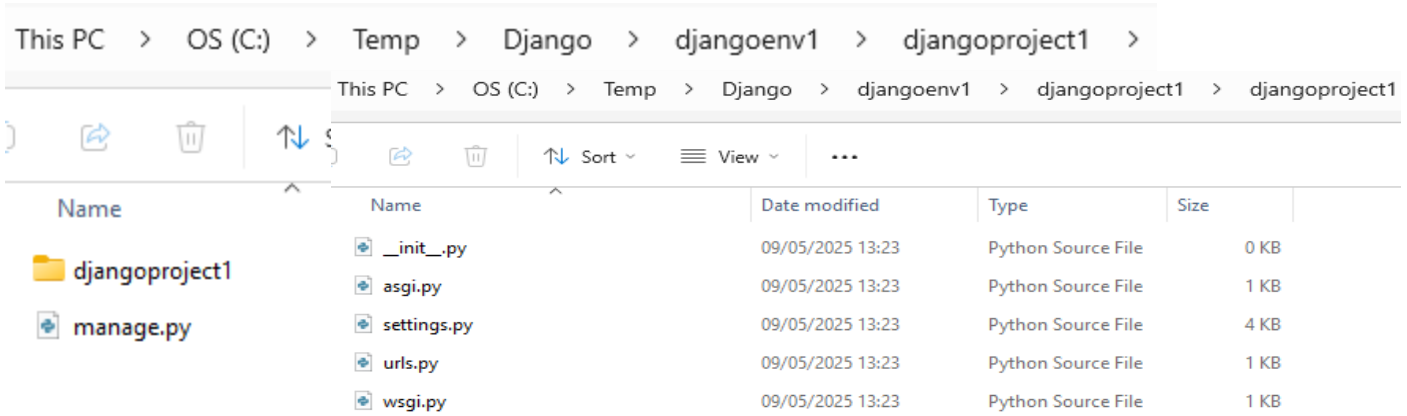
Create a Django Project

```
(djangoenv1) C:\Temp\Django\djangoenv1>django-admin startproject djangoproject1
```



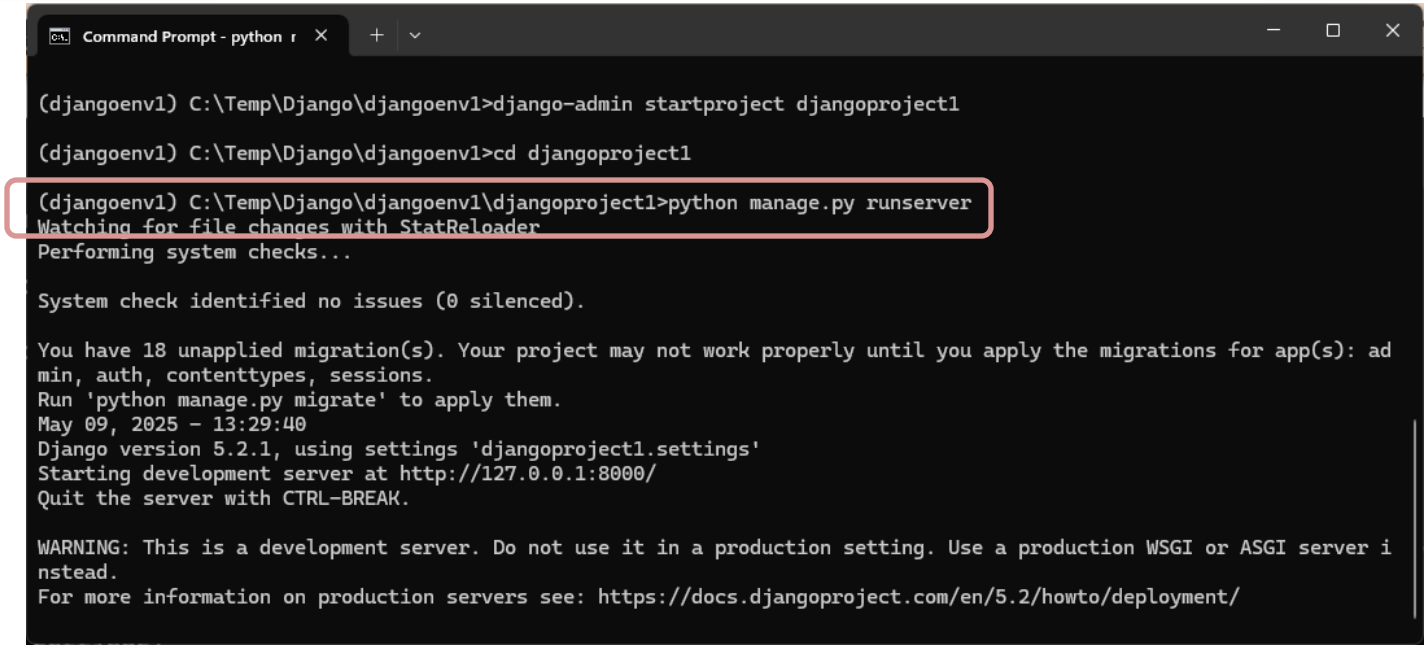
This command will create a Django Project called “djangoproject1”

The following folders and files are created:



Run the Django Project

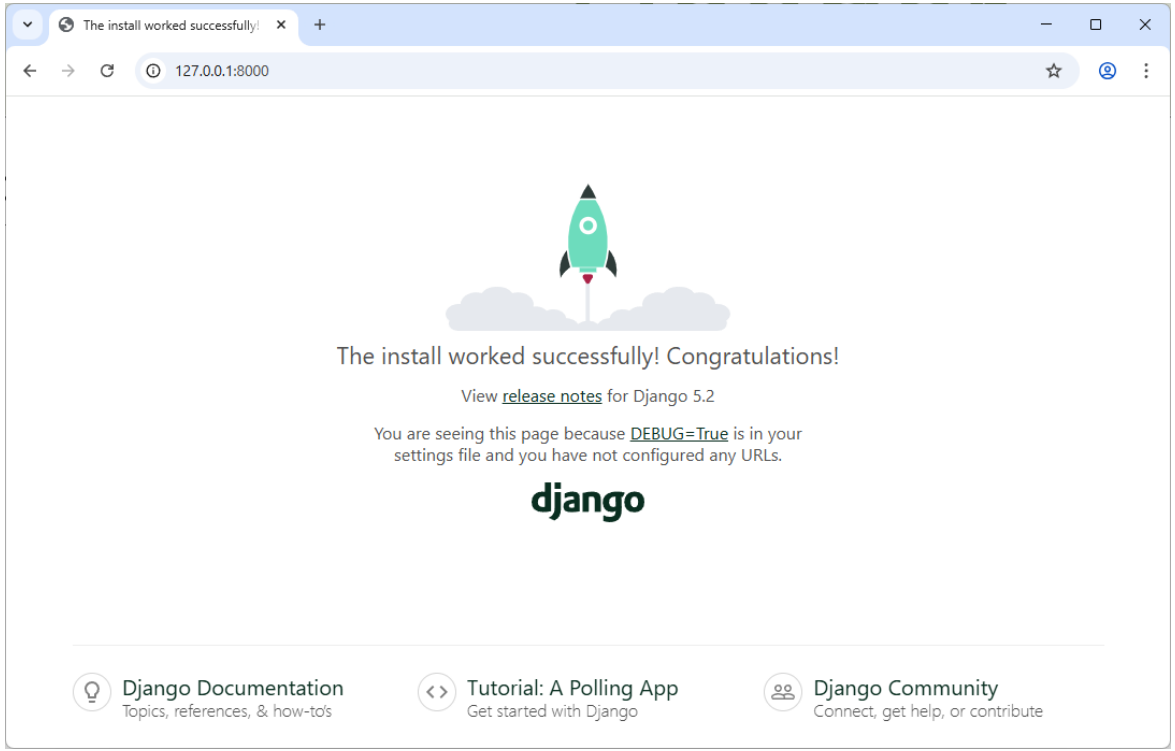
```
(djangoenv1) C:\Temp\Django\djangoenv1\djangoproject1>python manage.py runserver
```



Then open your Web Browser and enter the URL: **127.0.0.1:8000**

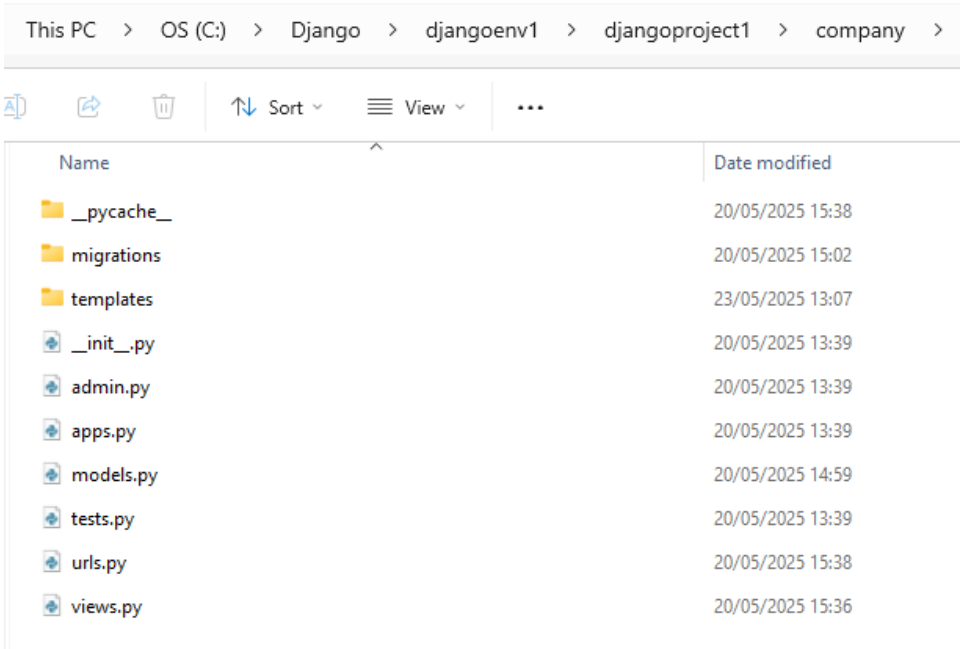
Run the Django Project

Open your Web Browser and enter the URL: **127.0.0.1:8000**



Create a **Django App**

```
(djangoenv1) C:\Temp\Django\djangoenv1\djangoproject1>python manage.py startapp company
```



This command will create a Django App called “company”.

https://www.halvorsen.blog

Django Views

Model - View - Template (MVT)

Table of Contents

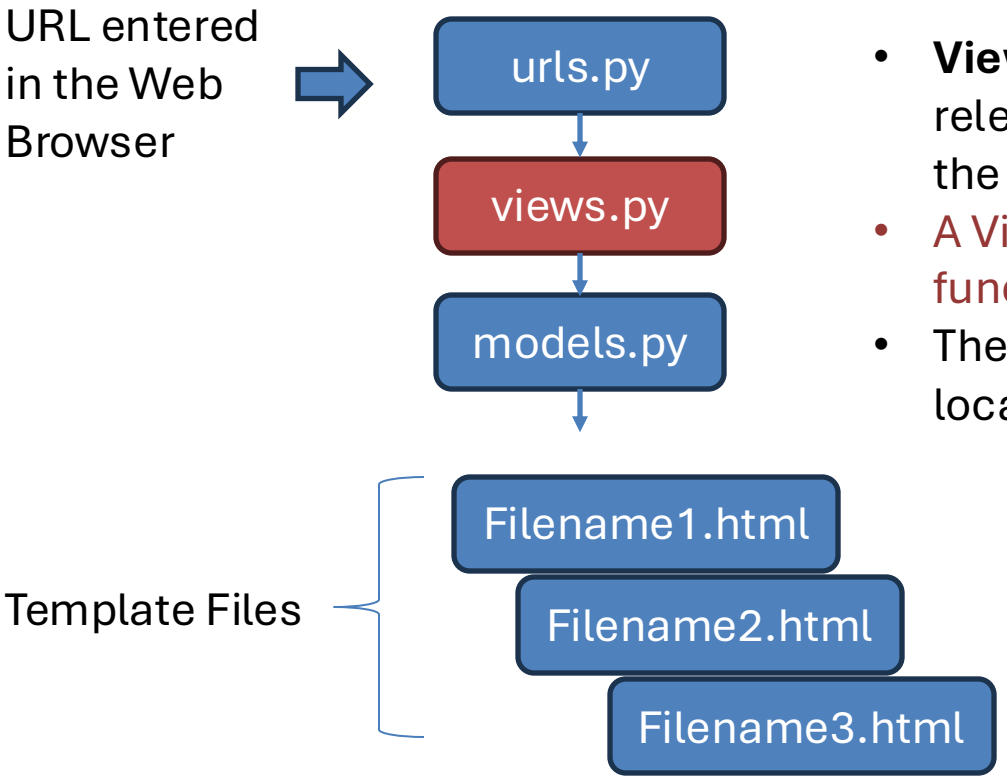
Hans-Petter Halvorsen

Model View Template (MVT)

Django follows the MVT design pattern (Model View Template).

- **Model** - The data you want to present, usually data from a database. The models are usually located in a file called “models.py”.
- **View** - A request handler that returns the relevant template and content - based on the request from the user. The views are usually located in a file called “views.py”.
- **Template** – A HTML file containing the layout of the web page, with logic on how to display the data. The HTML template files are stored in a subfolder called “**templates**”

Django Files



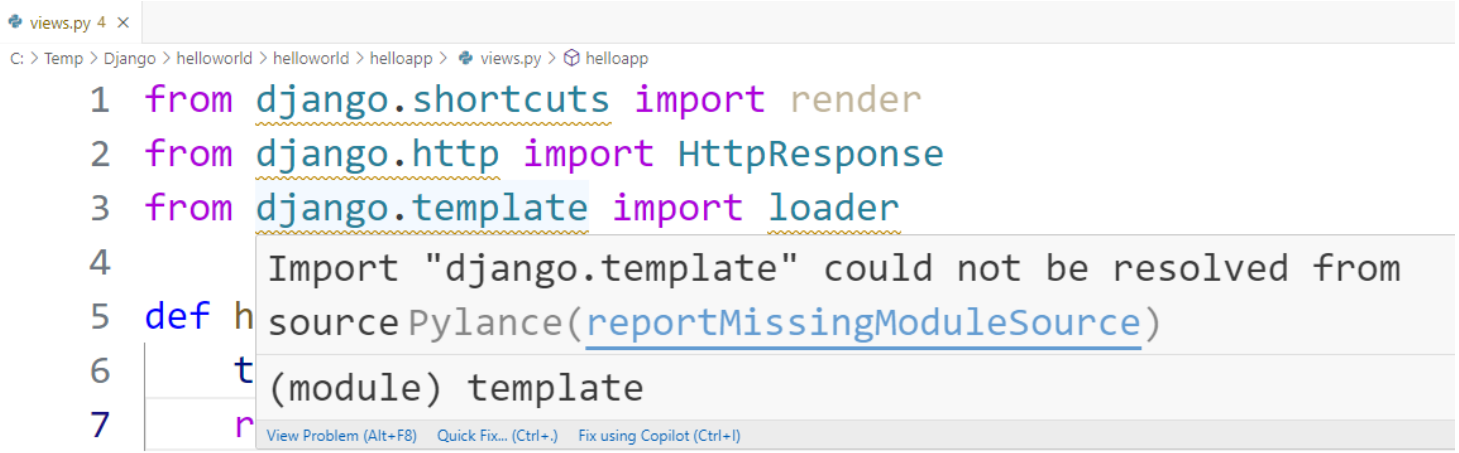
- **View** - A request handler that returns the relevant template and content - based on the request from the user.
- A View in Django is basically a Python function that handles an HTTP request.
- The Views (Python Functions) are usually located in a file called "**views.py**".

views.py

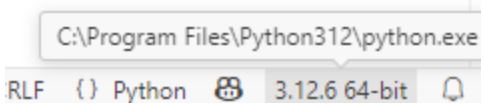
```
from django.shortcuts import render
from django.http import HttpResponse

def company(request):
    return HttpResponse("Hello World.")
```

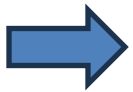
Errors in Visual Studio Code?

Do you get some Error/Warnings like this in Visual Studio Code:



This is because VS Code uses your "main" Python interpreter, whereas you should use the one in your virtual environment for Django.





Change the Python Interpreter

urls.py

“urls.py” for “company” Django App:

```
from django.urls import path
from . import views

urlpatterns = [
    path('company/', views.company, name='company'),
]
```

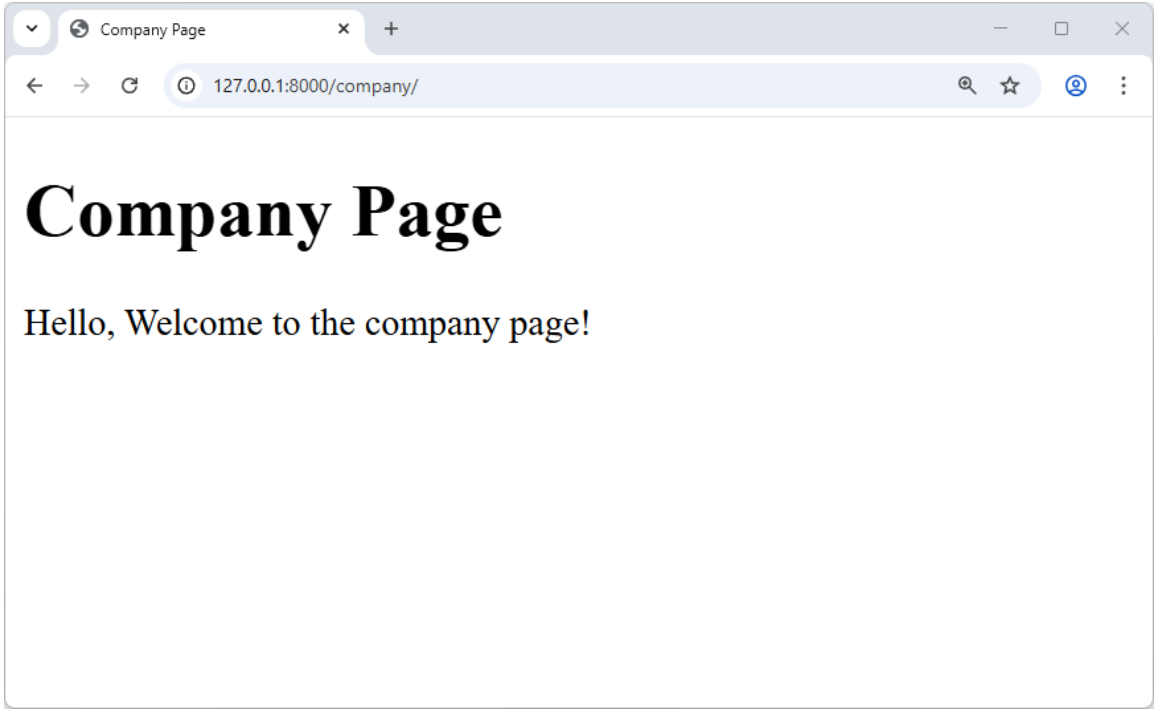
“urls.py” for “djangoproject1” Django Project:

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('company.urls')),
    path('admin/', admin.site.urls),
]
```

Run App “company”

```
(djangoenv1) C:\Temp\Django\djangoenv1\djangoproject1>python manage.py runserver
```

Then open your Web Browser and enter the URL: **127.0.0.1:8000/company**



https://www.halvorsen.blog

Django Templates

Model - View - Template (MVT)

Hans-Petter Halvorsen

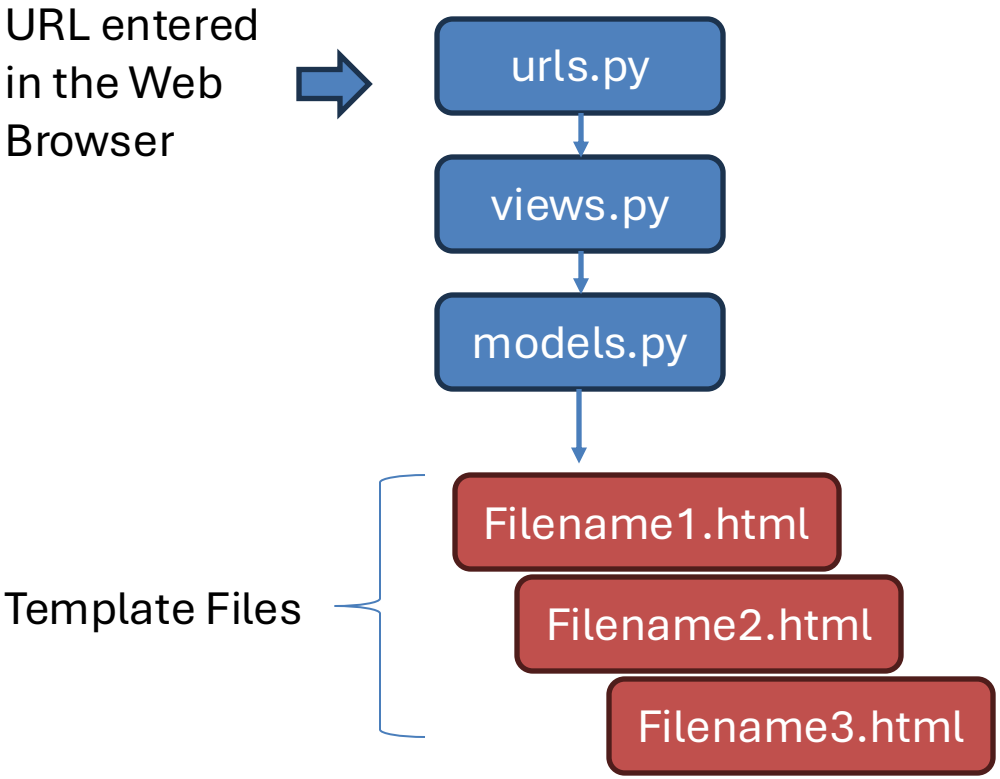
Table of Contents

Model View Template (MVT)

Django follows the MVT design pattern (Model View Template).

- **Model** - The data you want to present, usually data from a database. The models are usually located in a file called “models.py”.
- **View** - A request handler that returns the relevant template and content - based on the request from the user. The views are usually located in a file called “views.py”.
- **Template** – A HTML file containing the layout of the web page, with logic on how to display the data. The HTML template files are stored in a subfolder called “**templates**”.

Django Files



- Django Templates are HTML files containing the layout of the web pages, with logic on how to display the data
- The HTML template files are stored in a subfolder called “**templates**”

Templates

“**company.html**” Template:

```
<!DOCTYPE html>
<html>
<body>
    <h1>Company page</h1>
    <p>Hello, Welcome to the company page!>
</body>
</html>
```

We update “**views.py**”:

```
from django.shortcuts import render
from django.http import HttpResponse
from django.template import loader

def company(request):
    template = loader.get_template(company.html')
    return HttpResponse(template.render())
```

Settings (settings.py)

Then we need to modify the “settings.py” file:

settings.py

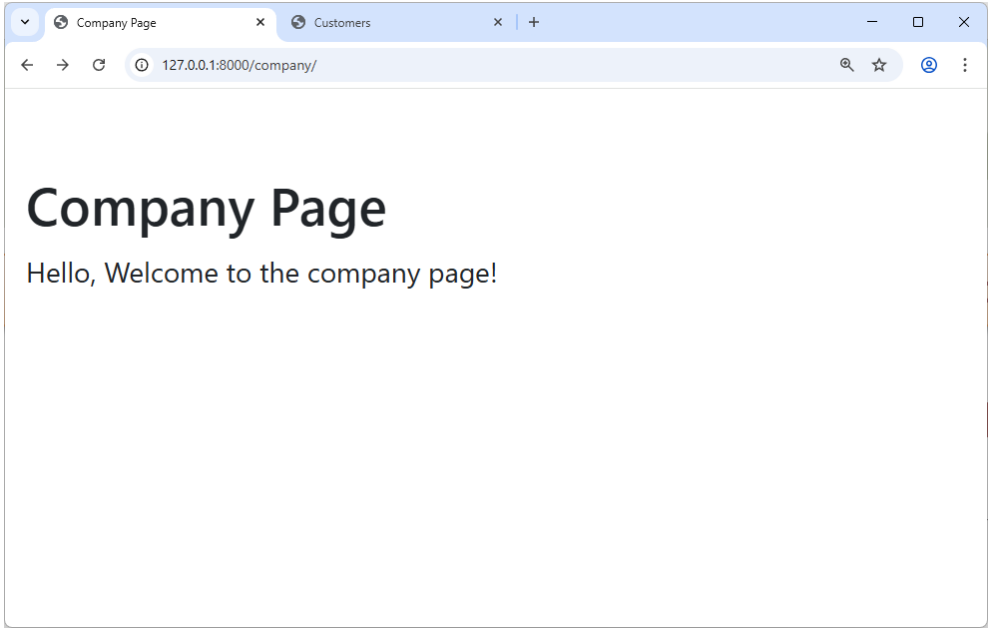
djangoproject1 > settings.py > ...

```
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'company'
41  ]
```

Run App “company”

```
(helloworld) C:\Temp\Django\djangoenv1\djangoproject1>python manage.py runserver
```

Then open your Web Browser and enter the URL: **127.0.0.1:8000/company**



https://www.halvorsen.blog

Django Models

Model - View - Template (MVT)

Hans-Petter Halvorsen

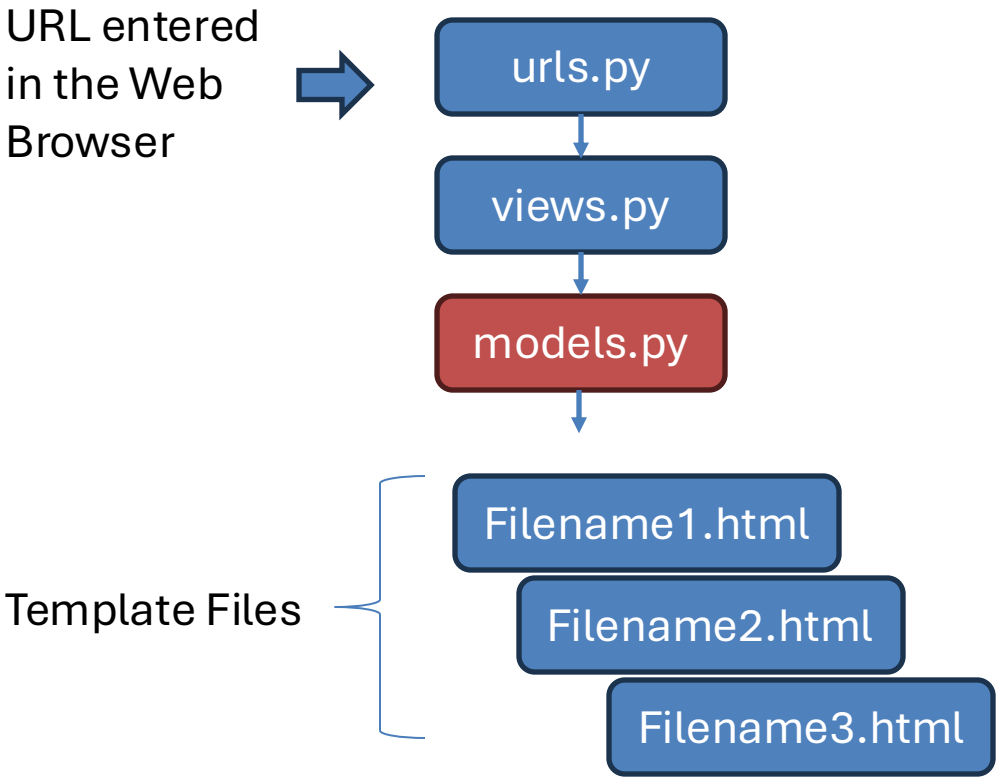
Table of Contents

Model View Template (MVT)

Django follows the MVT design pattern (Model View Template).

- **Model** - The data you want to present, usually data from a database. The models are usually located in a file called “models.py”.
- **View** - A request handler that returns the relevant template and content - based on the request from the user. The views are usually located in a file called “views.py”.
- **Template** – A HTML file containing the layout of the web page, with logic on how to display the data. The HTML template files are stored in a subfolder called “**templates**”

Django Files



- **Model** - The data you want to present, usually data from a database.
- The models are usually located in a file called "**models.py**"
- A Django model is basically a table in your database.

Django and Databases

- Django has support for databases like SQLite, MySQL/MariaDB and PostgreSQL.
- In Django, the data is delivered as an Object Relational Mapping (ORM), which is a technique designed to make it “easier” to work with databases without using complex SQL statements. It’s a bit like the “Entity Framework” used in C#.
- Django includes an SQLite database and has built-in support for this.
- For the others you need to install the database system you want to use from their respective websites.
  - In addition, you need to install a Python package/driver for the specific database system.
- For small applications SQLite is good enough, but for larger applications with lots of data and many users’ a database like MySQL/MariaDB or PostgreSQL should be used.

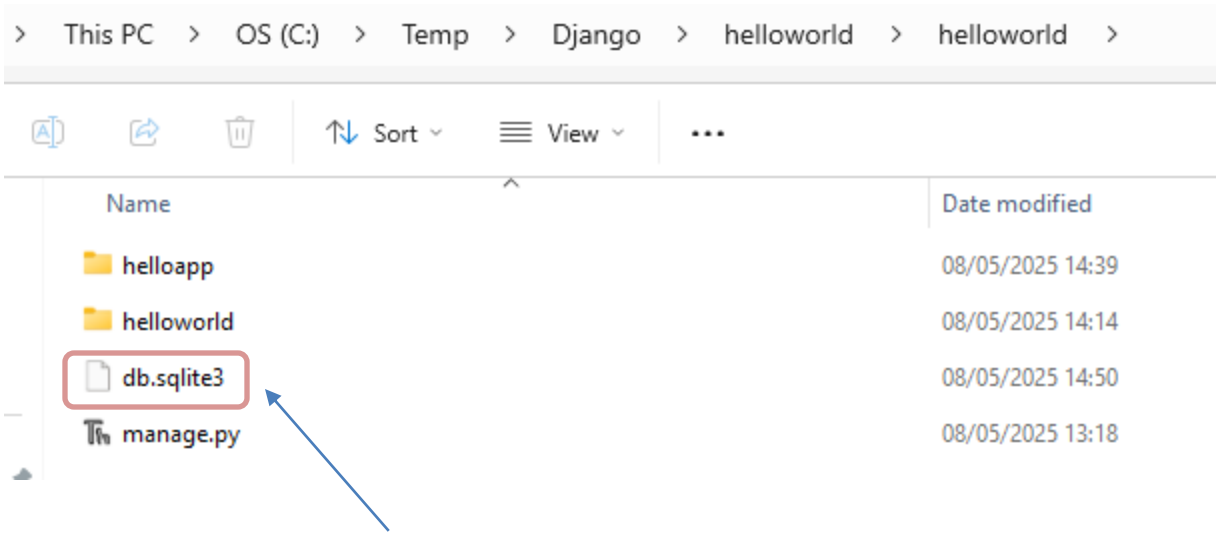
SQLite Database

Hans-Petter Halvorsen

Table of Contents

SQLite Database

Django includes an SQLite database, but for the others you need to install the database system you want to use from their respective websites and in addition install a Python package/driver for the specific database system.

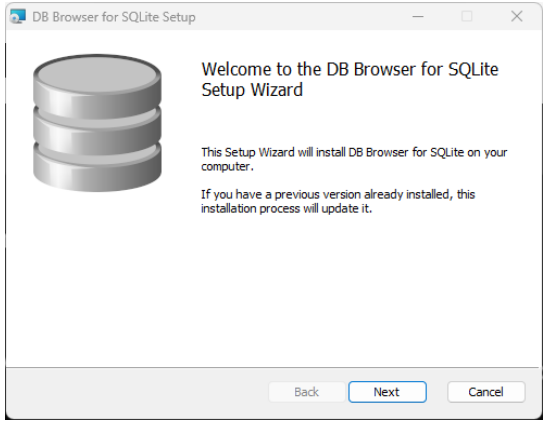


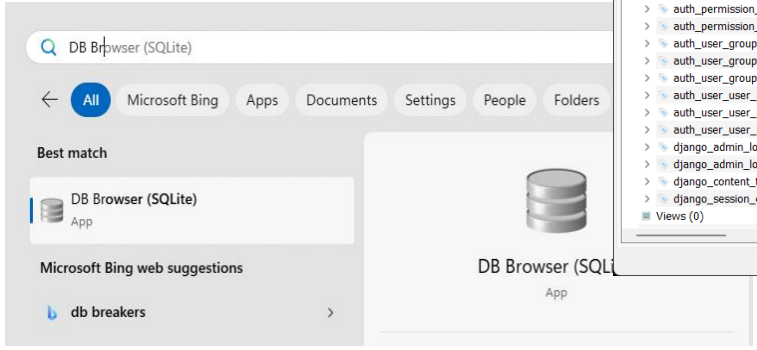
As part of the Django installation, you get an SQLite Database

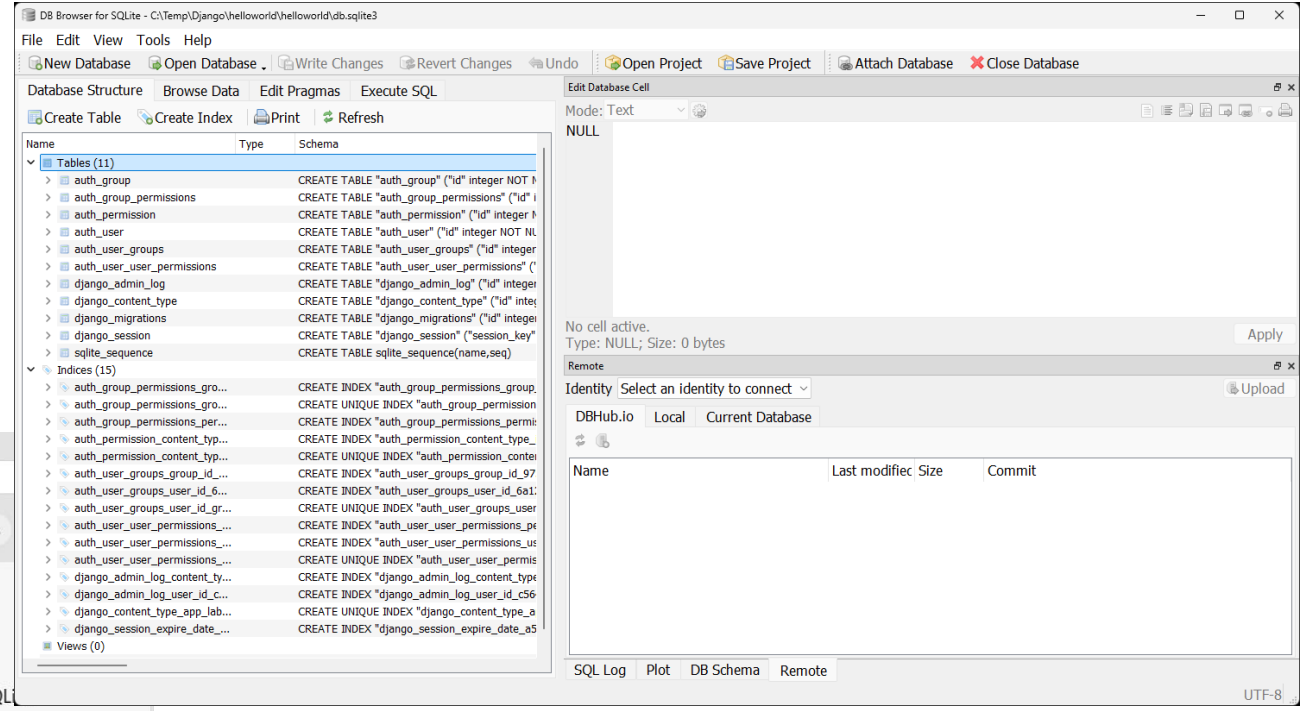
DB Browser for SQLite

You can download and install the free DB Browser for SQLite for creating Tables, perform SQL Queries, etc.

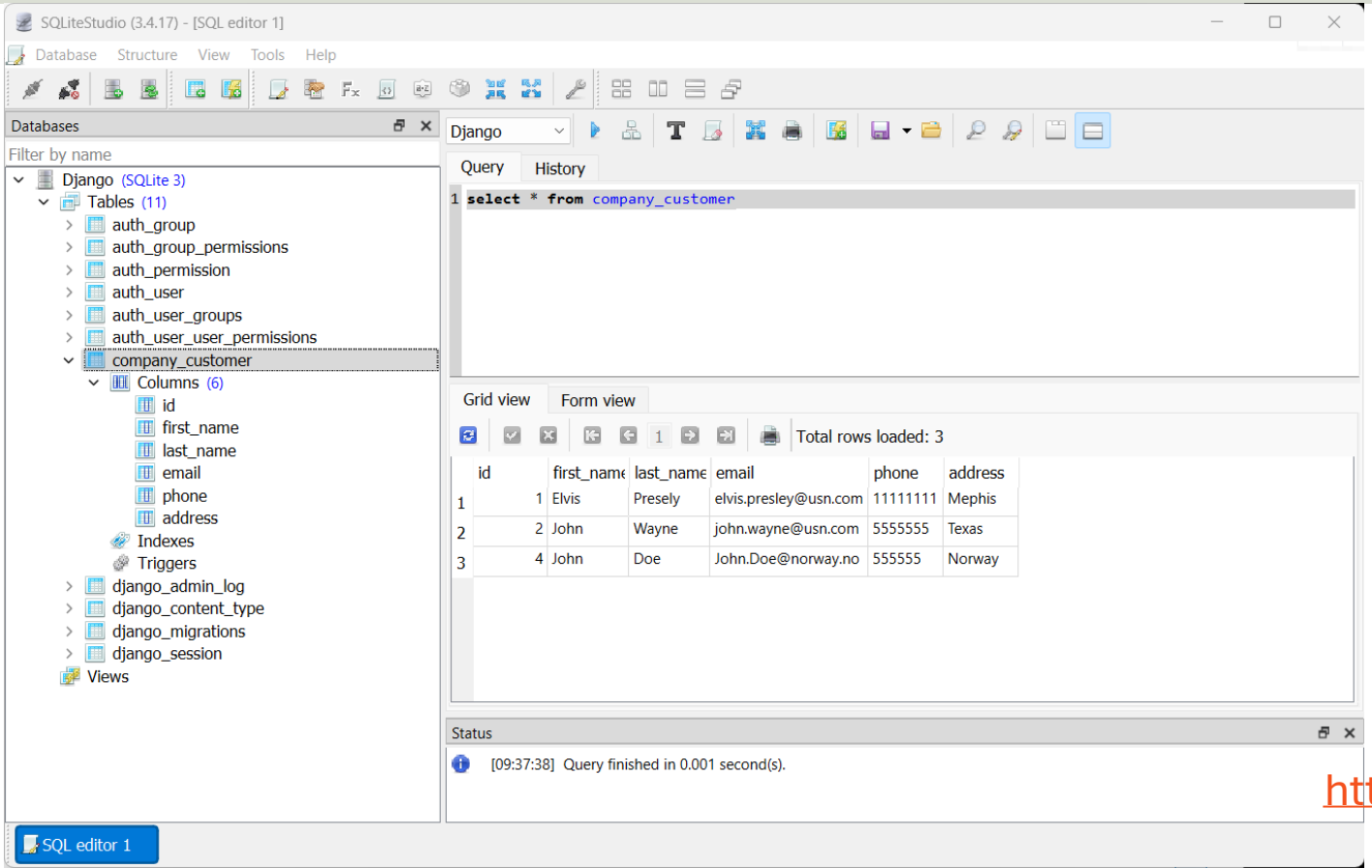
https://sqlitebrowser.org







SQLiteStudio



https://sqlitestudio.pl

Create the **Model** (Table)

Let's create a “Customer” Model (Table).We start by creating the Model in “**models.py**”:

models.py

company > models.py > ...

```python
from django.db import models

# Create your models here.
class Customer(models.Model):
    first_name  = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    email = models.EmailField()
    phone = models.CharField(max_length=15)
    address = models.TextField()

    def __str__(self):
        return self.first_name + ' ' + self.last_name
```

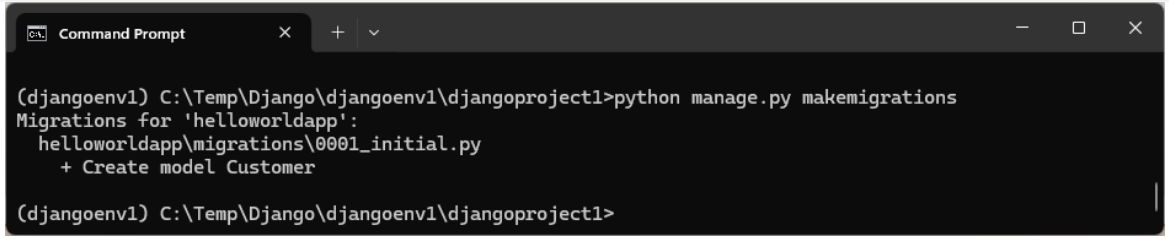
By default, all Models created in the Django project will be created as tables in the default SQLite database (“db.sqlite3”) that is installed with Django.

models.py

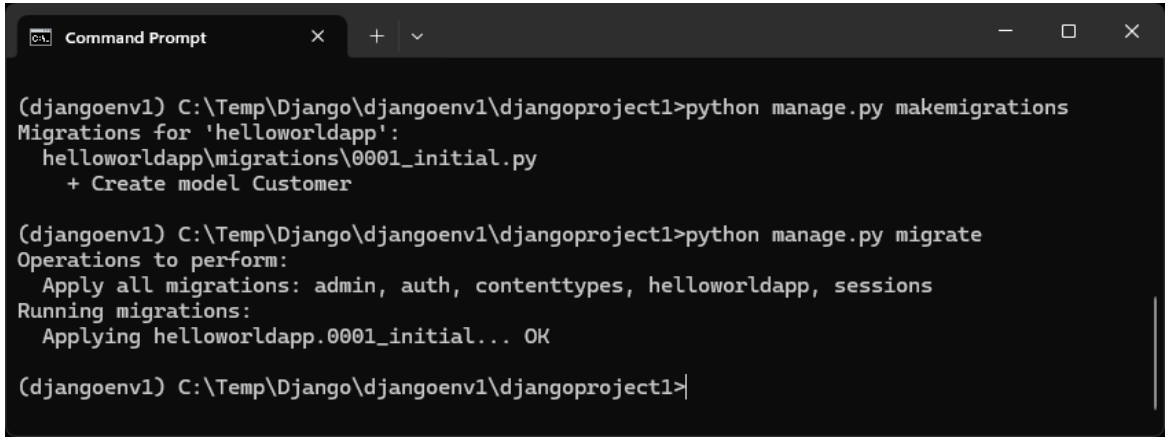
```
from django.db import models

# Create your models here.
class Customer(models.Model):
    first_name  = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    email = models.EmailField()
    phone = models.CharField(max_length=15)
    address = models.TextField()

    def __str__(self):
        return self.first_name + ' ' + self.last_name
```

Migrate

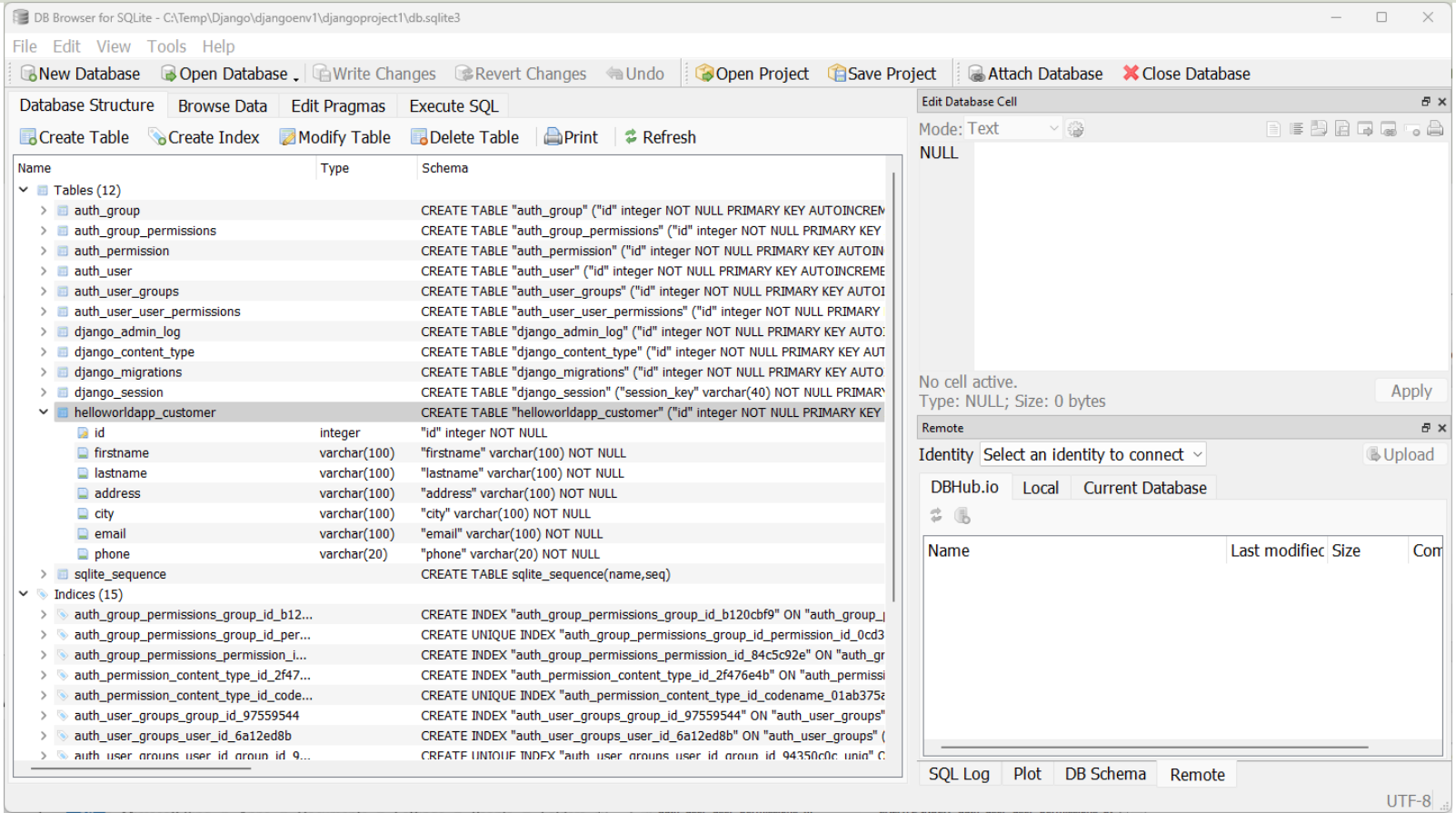
```
(djangoenv1) C:\Temp\Django\djangoenv1\djangoproject1>python manage.py makemigrations
```




```
(djangoenv1) C:\Temp\Django\djangoenv1\djangoproject1>python manage.py migrate
```



DB Browser for SQLite



Insert Data

You can insert data using Python. To open a Python shell:

```
(djangoenv1) ...>python manage.py shell
```

```
>>> Customer.objects.all().values()
<QuerySet []>
```

We see there is no data

WE can insert data like this:

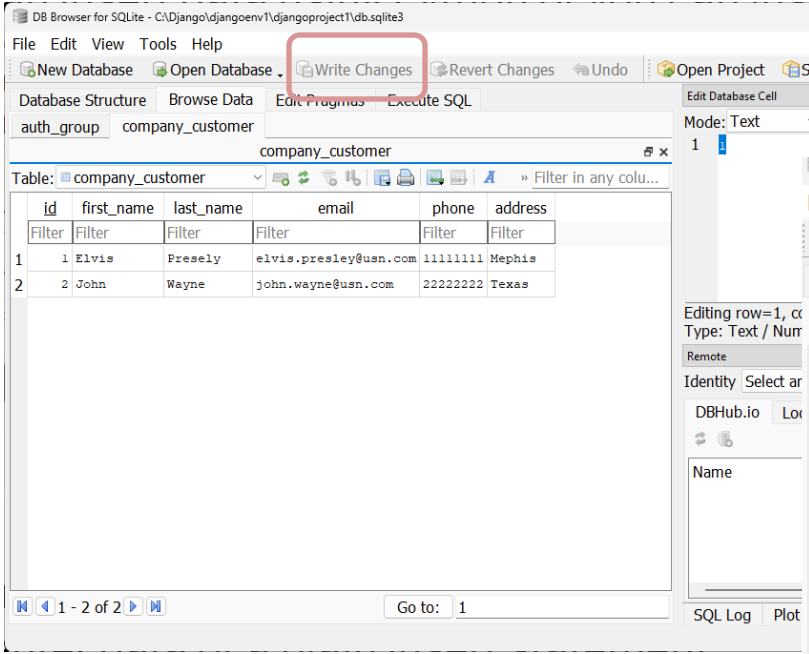
```
>>> customer = Customer(first_name='Hans-Petter', last_name='Halvorsen’,
email='hph@usn.no', phone='12345678', address='Norway')
>>> customer.save()
```

Then we can look for the data:

```
>>> Customer.objects.all().values()
<QuerySet [{'id': 1, 'first_name': 'Hans-Petter', 'last_name': 'Halvorsen',
‘email': 'hph@usn.no', 'phone': ‘12345678', 'address': 'Norway'}]>
```

Insert Data using DB Browser

You can insert data using Python or you can insert data using DB Browser.



Note! Make sure to click “Write Changes” to update the Database!!!

Note! The Table name is actually “company_customer”

In DB Browser you can use the GUI to enter data or a plain insert statement.

https://www.halvorsen.blog

Template and View for Customer Model

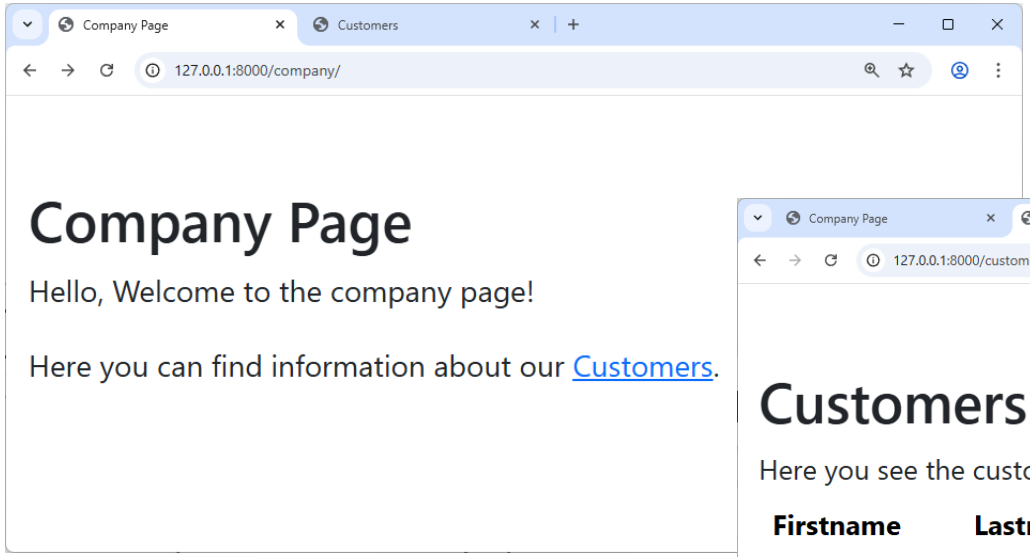
Model - View - Template (MVT)

Hans-Petter Halvorsen

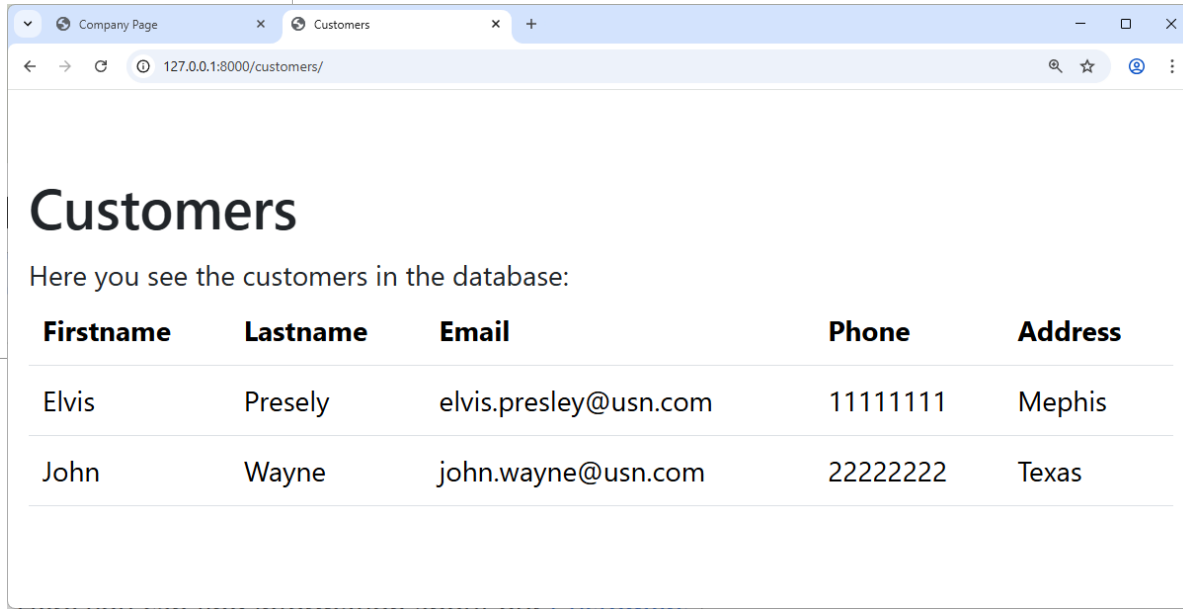
Table of Contents

Django “Company” App

Main “Company” App:



Subpage showing the “Customers” for the “Company” App:



Template and View for Customer Model

- We have created a Customer Model in Django
- Next, we need to create:
  - A new Template, we can call it “customers”
  - A new View, we can call it “customers”

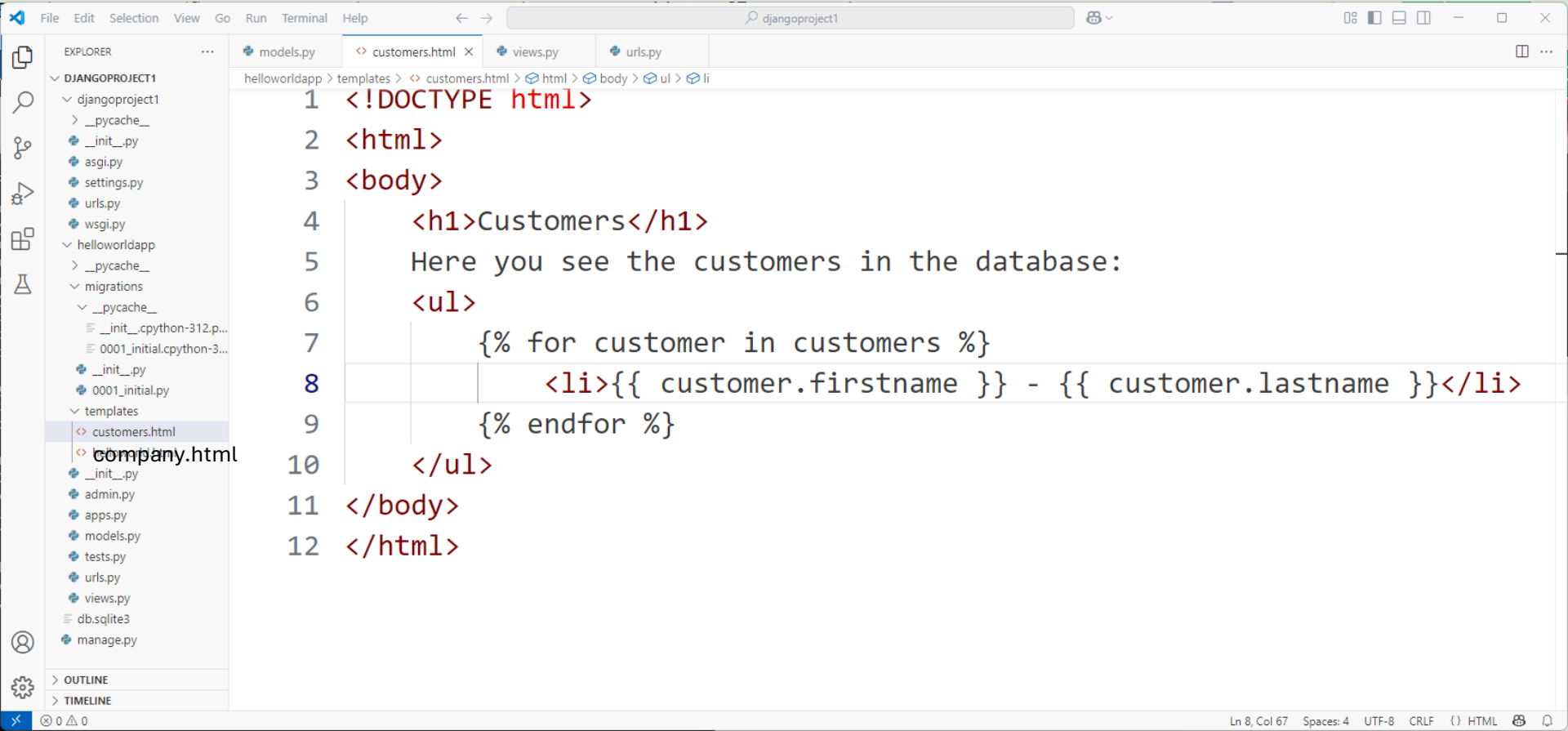
Create “customers” Template

Model - View - Template (MVT)

Table of Contents

Hans-Petter Halvorsen

Create the **Template**



customers.html Template

```
<!DOCTYPE html>
<html>
<body>
    <h1>Customers</h1>
    Here you see the customers in the database:
    <ul>
        {% for customer in customers %}
            <li>{{ customer.firstname }} - {{ customer.lastname }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

Note how we use Python code as part of the HTML file to present data from the database. The code inside **{% ... %}** is called **Django/Template Tags**.

Django/Template Tags

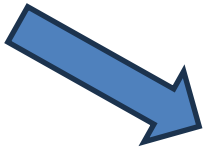
- Django or Template Tags are basically Python code that is integrated into the Template/HTML page.
- These tags are inside {% %} to separate it from the HTML tags.
- The Template Tags are executed server-side before the HTML page is sent to the client.

Django/Template Tags

```
<!DOCTYPE html>
<html>
<body>
    <h1>Customers</h1>
    Here you see the customers in the database:
    <ul>
        {% for customer in customers %}
            <li>{{ customer.firstname }} - {{ customer.lastname }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

The Template Tags are executed server-side before the HTML page is sent to the client.
You can use if-else, for loop, etc.

Template page (HTML page) Server-side



```
<!DOCTYPE html>
<html>
<body>
    <h1>Customers</h1>
    Here you see the customers in the database:
    <ul>
            <li>Hans-Petter - Halvorsen</li>
            <li>Elvis - Presley</li>
    </ul>
</body>
</html>
```

Resulting HTML page on the Client-side

Create “customers” View

Model - View - Template (MVT)

Hans-Petter Halvorsen

Table of Contents

views.py

```
from django.http import HttpResponse
from django.template import loader
from .models import Customer

def company(request):
    template = loader.get_template('company.html')
    return HttpResponse(template.render())

def customers(request):
    customers = Customer.objects.all()
    template = loader.get_template('customers.html')
    context = {
        'customers': customers,
    }
    return HttpResponse(template.render(context, request))
```

urls.py

Update “urls.py” for “company” Django App:

urls.py

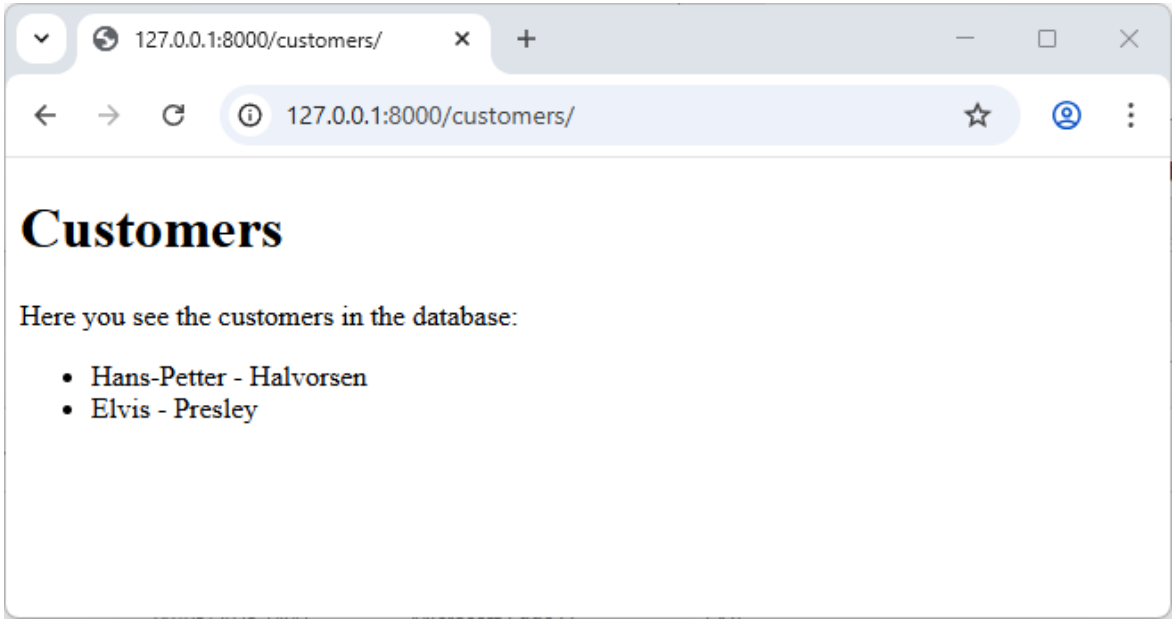
company > urls.py > ...

```python
from django.urls import path
from . import views

urlpatterns = [
    path('company/', views.company, name='company'),
    path('customers/', views.customers, name='customers'),
]
```

Run “customers” page

```
(djangoenv1) C:\Temp\Django\djangoenv1\djangoproject1>python manage.py runserver
```

Then open your Web Browser and enter the URL: **127.0.0.1:8000/customers**



https://www.halvorsen.blog

Improve UX in Django App with Bootstrap

Hans-Petter Halvorsen

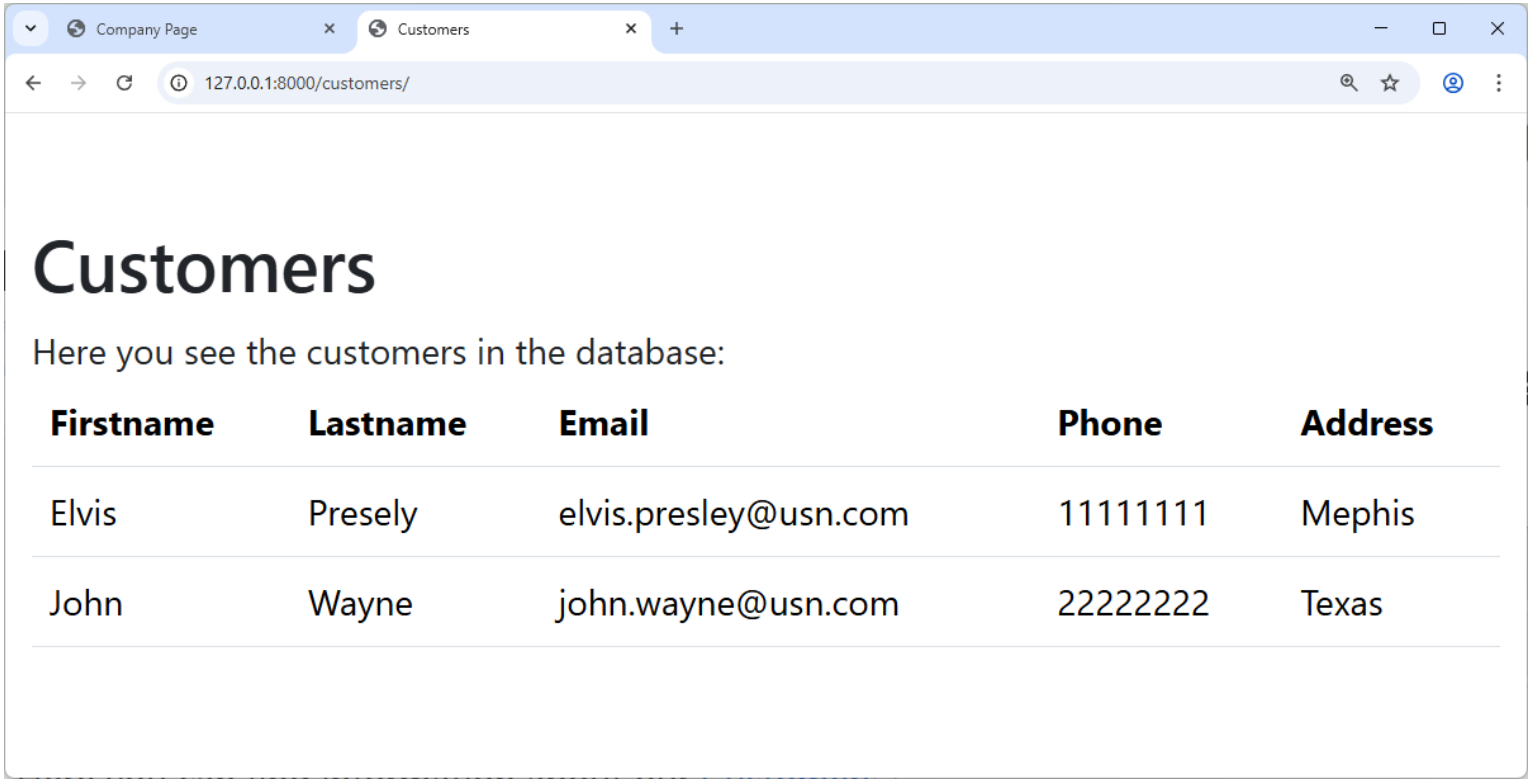
Table of Contents

Improve UX with Bootstrap

- Bootstrap is a free and open-source CSS framework.
- To create modern web pages and web applications today, it needs good design, layout and it needs to work on all kind of devices like PCs and smartphones, etc.
- Bootstrap is a popular CSS framework that is used by many web developers today. Bootstrap also uses JavaScript to create interactivity and animations, etc.
- By using such a framework like Bootstrap or similar you can focus on developing the application features and let Bootstrap do most of the layout, etc.
- https://getbootstrap.com

New “customers” page

We will put the data into an HTML table and use Bootstrap:



Using Bootstrap

You can start using Bootstrap in different ways. The simplest method is to include the CSS and the JavaScript libraries in in the <head></head> section your HTML files.

1. Put Bootstrap CSS Library:

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
```

2. Bootstrap JavaScript Library :

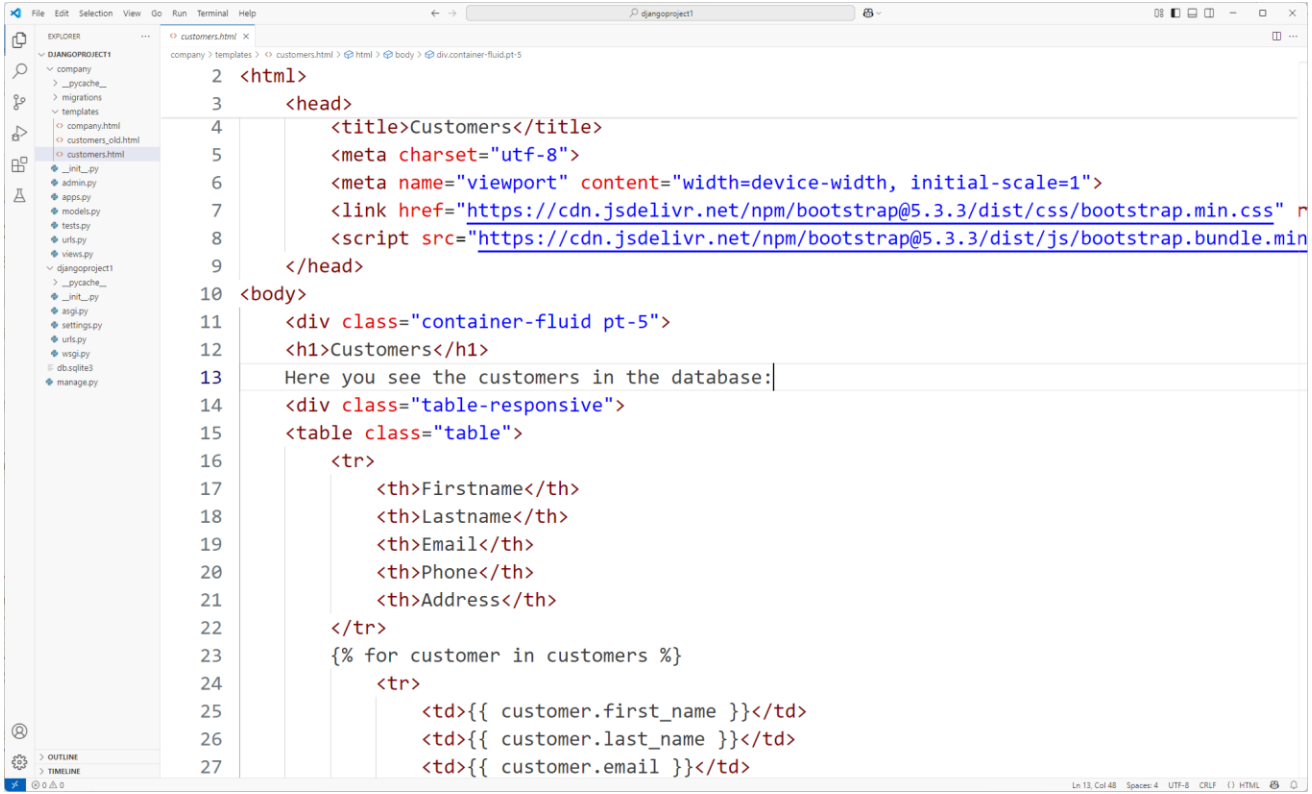
```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
```

3. This line of code ensures your code works fine on all devices like smartphones, etc.:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Update Template “customers.html”

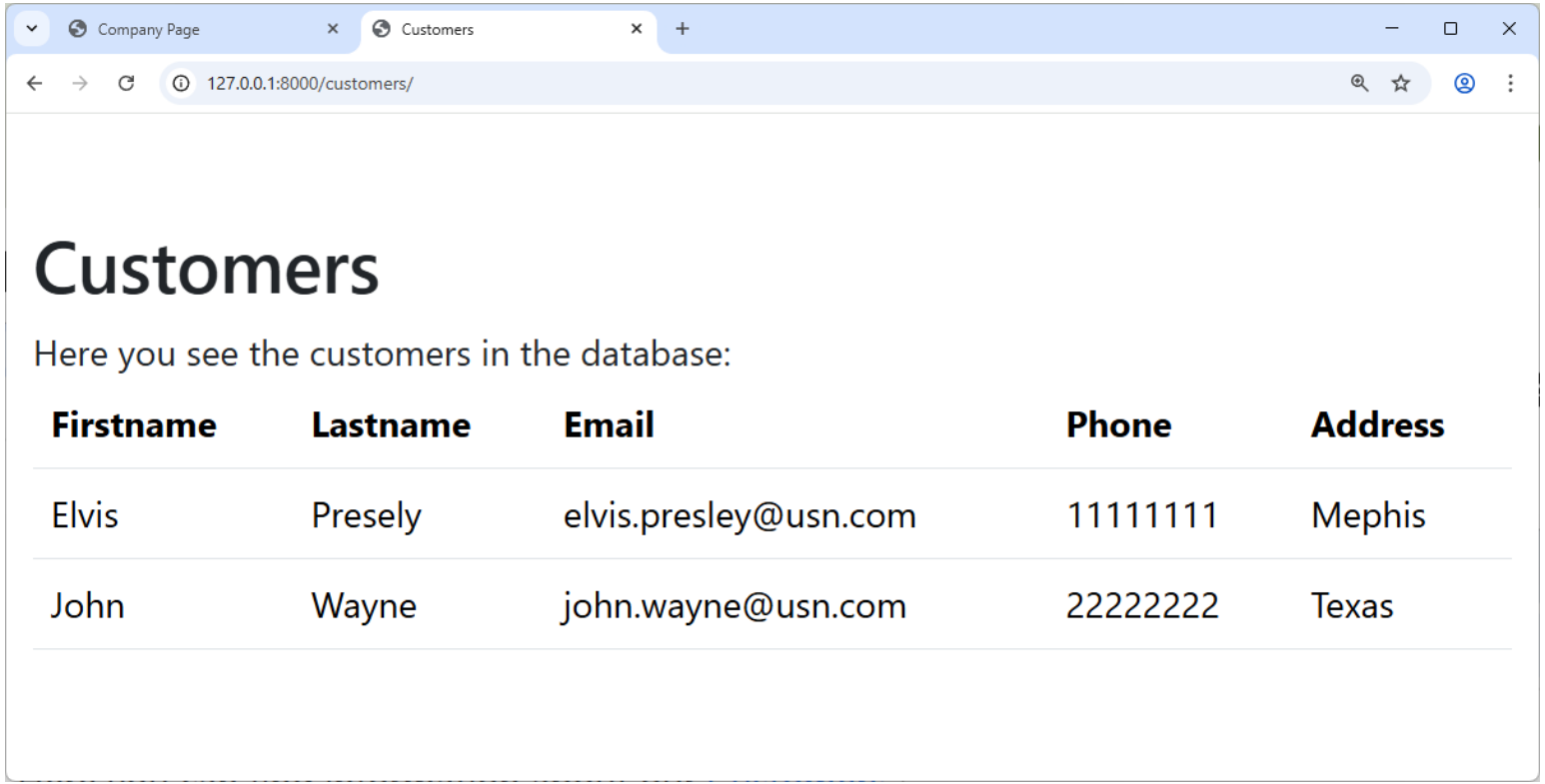
We typically want to present the data from the database using an HTML Table.



Customers.html

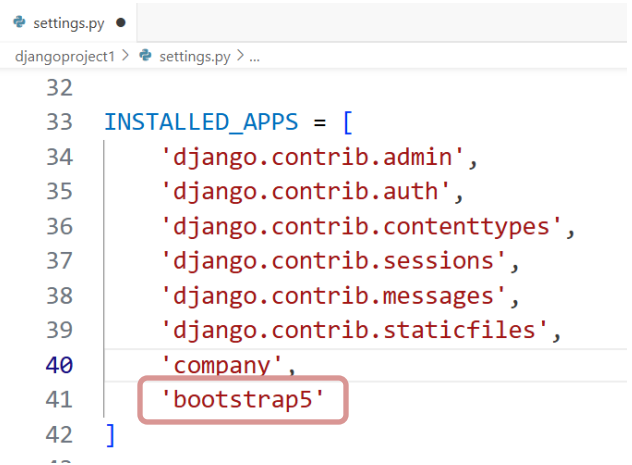
```
<!DOCTYPE html>
<html>
    <head>
        <title>Customers</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
    </head>
<body>
    <div class="container-fluid pt-5">
    <h1>Customers</h1>
    Here you see the customers in the database:
    <div class="table-responsive">
    <table class="table">
        <tr>
            <th>Firstname</th>
            <th>Lastname</th>
            <th>Email</th>
            <th>Phone</th>
            <th>Address</th>
        </tr>
        {% for customer in customers %}
            <tr>
                <td>{{ customer.first_name }}</td>
                <td>{{ customer.last_name }}</td>
                <td>{{ customer.email }}</td>
                <td>{{ customer.phone }}</td>
                <td>{{ customer.address }}</td>
            </tr>
        {% endfor %}
    </table>
</body>
</html>
```

Customers page with Bootstrap



Alternative Method

- An alternative way of using Bootstrap in Django is to install the Bootstrap in the Virtual Python Environment using PIP: pip install django-bootstrap-v5
- Then we need to include Bootstrap in “settings.py”



Then start using Bootstrap

Then we include Bootstrap like this in the Template files:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Customers</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        {% load bootstrap5 %}
        {% bootstrap_css %}
        {% bootstrap_javascript %}
    </head>
<body>
   ..
</body>
</html>
```

Basically, we need to add the following in the <head> section:

```
{% load bootstrap5 %}
{% bootstrap_css %}
{% bootstrap_javascript %}
```

https://www.halvorsen.blog

Tips and Tricks

Hans-Petter Halvorsen

Table of Contents

Multiple Apps in a Project?

- When Django looks for a template referred to in the render function, it uses the first file that matches the relative path.
- If you have multiple Django apps in the same project with same folder structures for templates, it's likely that one app can unintentionally use a template from another app.
- To avoid such errors, always create a subfolder under an app's templates folder that matches the name of the app to avoid any duplication.

app1/
 /templates/app1/
 ..
app2/
 /templates/app2/
 ..

Static Files

- Typically, you need to include and use static files like images, CSS files and JavaScript files
- Such Files should typically be put in a folder called “static” inside your App folder.
- The “static” folder is not created by default, so you need to manually create it.
- Note if DEBUG = False and steps are also necessary (se sources below for more information).
- Sources:
  - https://docs.djangoproject.com/en/5.2/howto/static-files/
  - https://www.w3schools.com/django/django_add_static_files.php

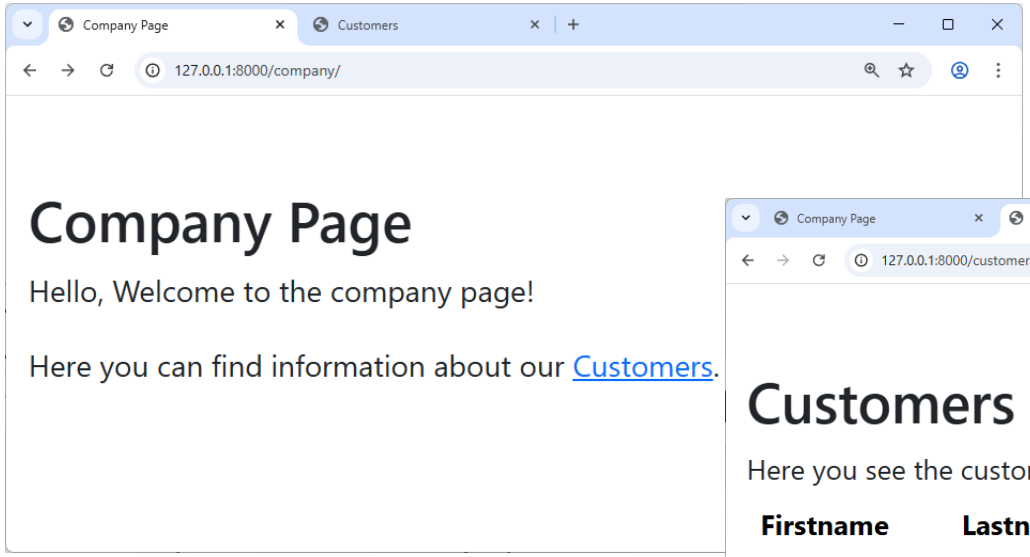
Static Files

Here is an example of how you can include and use a CSS file in your HTML Template:

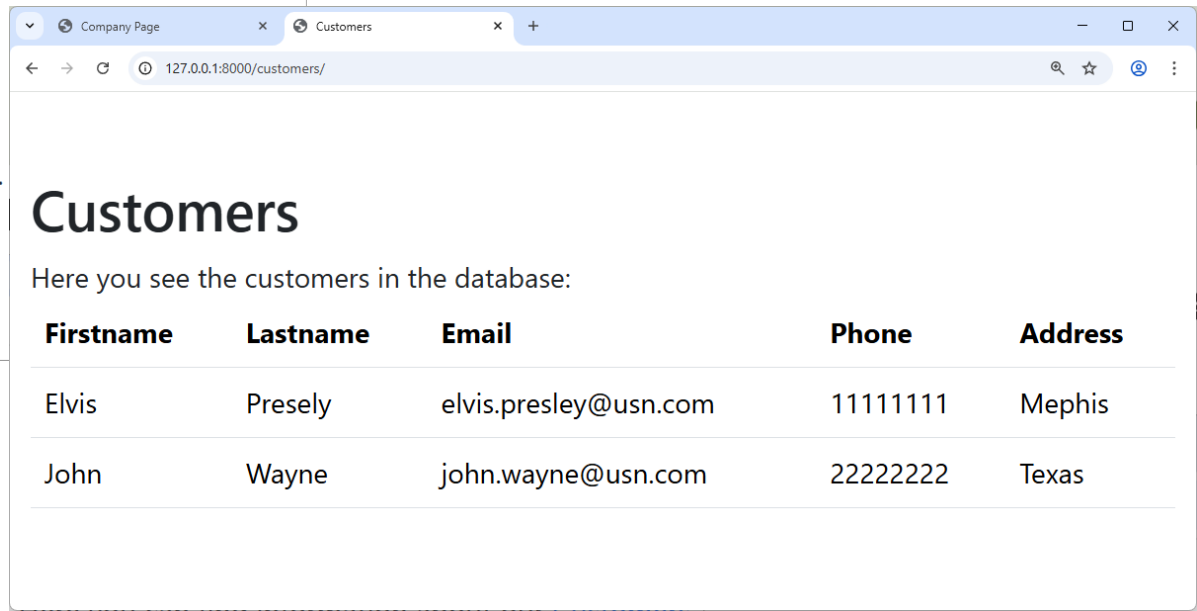
```
{% load static %}
<!DOCTYPE html>
<html>
<link rel="stylesheet" href="{% static 'myfirst.css' %}">
<body>
..
..
..
</body>
</html>
```

Final “Company” App

Main “Company” App:



Subpage showing the “Customers” for the “Company” App:



Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog