#### https://www.halvorsen.blog





# Django and PostgreSQL

#### Hans-Petter Halvorsen



### Contents

- 1. Introduction
  - <u>Django</u>
  - PostgreSQL
- 2. Create Django Project and App
- 3. <u>Django + PostgreSQL</u>
- 4. Customer App
- 5. Django Admin

#### https://www.halvorsen.blog

## Introduction

#### Hans-Petter Halvorsen



### Introduction

- We will create a Django Project and a Django Web Application.
- 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.
- In this Tutorial we will see how we can use **Django** in combination with **PostgreSQL**.
- We will create a simple "**Customer App**" that retrieves a list of Customers stored in the PostgreSQL database.
- Finally, we will use the built-in "**Django Admin**" module to create **CRUD** functionality, i.e., the possibility to Create, Read, Update and Delete Customers in the PostgreSQL database.

### **Customer App**

We shall create a basic Django Web Application like this where the data are stored in a PostgreSQL database:

	rension nooso, customers,			~ 0
Custome	ers			
lere you see the cus	stomers in the database:			
Firstname	Lastname	Email	Phone	Address
Elvis	Presley	elvis.presley@usa.com	11111111	Memphis
John	Wayne	john.wayne@usa.com	22222222	Texas

### **Software and Tools**

- Python
  - We need to have Python installed to install and use Django.
- Django
  - Django is basically a Python package installed using PIP which is part of Python.
- PostgreSQL database
- Visual Studio Code

#### https://www.halvorsen.blog



# Django

#### Hans-Petter Halvorsen



### 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: <a href="https://www.djangoproject.com">https://www.djangoproject.com</a>

### 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**



### 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.

#### https://www.halvorsen.blog

## PostgreSQL

#### Hans-Petter Halvorsen





- PostgreSQL is an open-source relational database system.
- PostgreSQL exists for Windows, macOS and Linux.
- Homepage: <u>https://www.postgresql.org</u>
- EnterpriseDB (EDB) is the company that is one of the largest contributor to PostgreSQL and responsible for the installer.
- EDB offer paid services for enterprises, but PostgreSQL itself is free.
- ERD Download Page: <u>https://www.enterprisedb.com/downloads/postgres-postgresql-downloads</u>

### Installation

<table-of-contents></table-of-contents>		\min Setup		– 🗆 X	
PACKAGED BY	Setup - PostgreSQL	Select Components			
<b>EDB</b>	Welcome to the PostgreSQL Setup Wizard.	Select the components you want to install; dea you are ready to continue.	ar the components yo	ou do not want to install. Click Next when	
		<ul> <li>PostgreSQL Server</li> <li>pgAdmin 4</li> <li>Stack Builder</li> <li>Command Line Tools</li> </ul>	Click on a	component to get a detailed description	
$\sim$			i	Setup	– 🗆 X
				Password	<b>1</b>
PostgreSQL				Please provide a password for the database superuser Password Retype password	(postgres).
	< Back Next	InstallBuilder	Make	e sure to rememb	per the Password!
l just use the	e default installatio	on,			
you need to	create a password	l for the database	τ.		
suneruser t	hat you need to ren	nemher for later	Ir	Istandunuer	< Back Next > Cancel

### pgAdmin



pgAdmin is graphical tool for managing your PostgreSQL database. pgAdmin is part of the installer from EDB. Here you can create new Database, new Tables, execute SQL Scripts, etc.

Φ.

### **PostgreSQL** Connection

¶P p	gAdmin 4							- 0	×
File (	Object Tools Edi	t View Window Help							
T	Welcome	Students/postgres ×							
**									
¢			Let's connect to th	e server					
			Existing Server (Optional)	😡 Post	tgreSQL 17		x   ~		
		\$	Server Name	Postgre	SQL 17				
		Welcome to the Query Tool Workspace!	Host name/address	localhos	st				
		The Query Tool is a robust and	Port	5432					
		versatile environment designed for executing SQL commands and reviewing result sets	Database	postgre	S		x   ~		
		efficiently.	User	postgre	s		×   ~		
		In this workspace, you can seamlessly open and manage	Password	•••••			¢		
		multiple query tabs, making it easier to organize your work.	Role	Select a	n item		· •		
		You can select the existing servers or create a completely	Service						
		new ad-hoc connection to any database server as needed.	Connection Paramet	ters			+		
			Name		Keyword	Value			
			SSL mode	<b>v</b>	sslmode	prefer	x   ~		
			Connection tim		connect timeout	10			
					Ð R	eset 😽 Connect & Ope	en Query Tool		

The Password you entered during installation of the PostgreSQL database

#### https://www.halvorsen.blog

# Create Django Project and App

django

#### Hans-Petter Halvorsen

**Table of Contents** 

### 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**

Q cmd		Q terminal		
← All Microsoft Bing Apps Documents S	Settings People Folders 🕨 🥘 …	CAll Microsoft Bing Apps Documents	s Settings People Folders 🕨	
Best match				
Command Prompt System		Best match		
Microsoft Bing web suggestions	Command Prompt <sub>System</sub>	App	>_	
▶ cmd prompt >		iner (10.)	Torminal	
🖾 Command Prompt X + 🗸			Terminar	
:\Users\hansp>		Windows PowerShell Copyright (C) Microsoft Corporation. All Install the latest PowerShell for new fea	rights reserved. atures and improvements! https://a	.ka.ms/PSWindows
		PS C:\Users\hansp>		

#### Create and Activate Virtual Python Environment

	📁 Django	× 🔁 Development	× + ×				
	← → ↑ C [	> This PC > C	$c_{\rm S}(c_{\rm C}) \rightarrow T_{\rm ef}$ In this example the Virtual Python				
	⊕ New ~ 🔏 🗘 🚺	) () () ()	The Sort Environment is called "djangoenv"				
	☆ Home Gallery	Name	Date modified     Type     Size       15/05/2025 12:29     File folder				
	> 🌰 Hans-Petter - Personal		$\blacksquare$ Command Prompt X + $\checkmark$ - $\Box$ X				
>python -m venv dj	angoenv	*	Microsoft Windows [Version 10.0.26100.4061] (c) Microsoft Corporation. All rights reserved.				
	🞍 Downloads	*	C:\Users\hansp>cd \				
	🔀 Pictures	*	C:\>cd Temp/Development				
C:\\djangoenv\S	cripts> <mark>activa</mark>	te.bat	C:\Temp\Development>python -m venv djangoenv				
	helloworldapp		C:\Temp\Development>cd djangoenv				
	Django		C:\Temp\Development\djangoenv>cd Script The system cannot find the path specified.				
			C:\Temp\Development\djangoenv>cd Scripts				
	🗸 📮 This PC		C:\Temp\Development\djangoenv\Scripts>activate.bat				
	> 🛄 OS (C:)		(djangoenv) C:\Temp\Development\djangoenv\Scripts>				
	1 item						

### Install Django and Create Django Project

#### Install Django:



Create Django Project: ...>d

	>py	thon	-m	pip	inst	all	Dja	ing	0	
Com	mand Prompt	× +	~					—		×
(djangoenv) C:\Temp\Development\djangoenv>django-admin startproject company										
(django	env) C:\Temp\L	eve Lopmen	t\djan	goenv>						
ango	o-admin	star	tpr	oject	t com	ipan	У			

In this example the Django Project is called "company"

### Run the Django Project



### Create a Django App

Command Prompt X + V	-	· □ ×	In this example the D	jango
(djangoenv) C:\Temp\Development\djangoenv\company>python manage.	by startapp custome	ers	Project is called " <b>cus</b>	stomers"
(djangoenv) C:\Temp\Development\djangoenv\company>				
	Y         THe         Edit         Selection         View         Go         Ru           D         DPLOBEA	n terminal Help. ← →	<pre>Prompany &amp;- ION = 'company.wsgi.application'</pre>	
<pre>&gt;python manage.py startapp c</pre>	ustomers	72 # Database 73 # https://doc	s.djangoproject.com/en/5.2/ref/settings/	/#databases
	manage py	74 75 DATABASES = { 76 'default' 77 'ENGI 78 'NAME 79 } 80 } 81	: { NE': 'django.db.backends.sqlite3', ': BASE_DIR / 'db.sqlite3',	
We can then open the project in Visual Studio Code:	© → OUTUNE → TIMELINE	PROILENS OUTPUT DEBUG CONDUL <u>TERMINAL</u> PS C:\Temp\Development\djangoem\company> []	nons	jowennet ≜ +- □ ≋ ··· ^ ×

### Create a Django App

After creating the Django App, you need to insert the "customers" App in "settings.py":

🕏 settings.py 🗙	
company > 🍖 set	tings.py >
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	'customers'
41	]

# https://www.halvorsen.blog django Django + PostgreSQL

#### Hans-Petter Halvorsen

Table of Contents



#### We can create a new Database called "Company"

🗣 pg	JAdmin 4			-		×							
File C	Dbject Tools Edit View Window Help												
-	Object Explorer 🛛 🕃 🔠 🖬 🔍 💽					:							
8	✓												
	🗸 🊍 Databases (2)	🍮 Create - Database		×									
ين م	> Students	General Definition S	ecurity Parameters A	Advanced SQL	(B)	ng Admin 4					_		×
-0	> A Login/Group Roles	Database	Company		File	Object Tools Edit V	iew Window Help					U	~
	> 🏊 Tablespaces	OID			•18	Welcome 💲 Co	npany/postgre ×						
		Owner	🐣 postgres	v									
		Comment			5								
				li li	÷			Let's connect to t	he server				
							8	Existing Server (Optional)	PostgreSQL 17	x   ~			
							Welcome to the Query Tool Workspace!	Server Name	PostgreSQL 17				
							The Query Tool is a robust and versatile environment designed	Host name/address	localhost				
		00		🗙 Close 🕢 Reset 🕞 Save			for executing SQL commands and reviewing result sets	Port	5432				
							entcientiy.	Database	Company	x   ~			
\$							seamlessly open and manage multiple query tabs, making it	User	postgres	x   ~			
-							easier to organize your work. You can select the existing	Password	•••••				
							servers or create a completely new ad-hoc connection to any	Role	Select an item	~			

database server as needed.

Service

Reset

😪 Connect & Open Query Tool

 $\mathbf{a}$ 

### psycopg2

I order to use PostgreSQL with Django we need to install the "**psycopg2**" Python package in your virtual Python Environment where you are using Django.

#### >pip install psycopg2-binary

Command Prompt X + ~	-		×					
(djangoenv) C:\Temp\Development\djangoenv>pip install psycopg2-binary								
Downloading psycopg2_binary-2.9.10-cp312-cp312-win_amd64.whl.metadata (5.0 kB) Downloading psycopg2_binary-2.9.10-cp312-cp312-win_amd64.whl (1.2 MB)								
Installing collected packages: psycopg2-binary Successfully installed psycopg2-binary-2.9.10	0.00							
<pre>[notice] A new release of pip is available: 24.2 -&gt; 25.1.1 [notice] To update, run: python.exe -m pip installupgrade pip</pre>								
(djangoenv) C:\Temp\Development\djangoenv>								

### "settings.py"

TIMELINE

Then change the Database Connection Setting in "settings.py". In this file the default database is the SQLite database that comes preinstalled with Django.



IN powershell ▲ + ~ □ 💼

### Migrate

#### ..>python manage.py makemigrations

#### ..>python manage.py migrate

Command Prompt × +

-

(djangoenv) C: \Temp\Development\djangoenv\company>python manage.py make migrations No changes detected

(djangoenv) C:\Temp\Development\djangoenv\company>python manage.py migrate Operations to perform: Apply all migrations: admin, auth, contenttypes, sessions Running migrations:

Applying contenttypes.0001\_initial... OK Applying auth.0001\_initial... OK Applying admin.0001\_initial... OK Applying admin.0002\_logentry\_remove\_auto\_add... OK Applying admin.0003\_logentry\_add\_action\_flag\_choices... OK Applying contenttypes.0002\_remove\_content\_type\_name... OK Applying auth.0002\_alter\_permission\_name\_max\_length... OK

Applying auth.0004\_alter\_user\_username\_opts... OK Applying auth.0005\_alter\_user\_last\_login\_null... OK Applying auth.0006\_require\_contenttypes\_0002... OK Applying auth.0007\_alter\_validators\_add\_error\_messages... OK Applying auth.0008\_alter\_user\_username\_max\_length... OK Applying auth.0009\_alter\_user\_last\_name\_max\_length... OK Applying auth.0010\_alter\_group\_name\_max\_length... OK

Applying auth.0011\_update\_proxy\_permissions... OK Applying auth.0012\_alter\_user\_first\_name\_max\_length... OK Applying sessions.0001\_initial... OK

(djangoenv) C:\Temp\Development\djangoenv\company>

The migration process updates the PostgreSQL database.



Here we see the default Django database tables has been created:



#### https://www.halvorsen.blog

## **Customers** App

django

#### Hans-Petter Halvorsen

Table of Contents

### **Customer App**

We shall create a basic Django Web Application like this where the data are stored in the PostgreSQL database:

~	Customers	×	Select customer to change	Dja ×   +					-		×
←	→ C	127.0.0.1:8000/cus	stomers/						☆	0	:
Customers											
Here you see the customers in the database:											
Fi	rstname	Lastna	me Email				Phone	A	ddress		
E	vic	Procley	elvis p	reslev@usa.com			11111111	м	omphic		

Firstname	Lastname	Email	Phone	Address
Elvis	Presley	elvis.presley@usa.com	11111111	Memphis
John	Wayne	john.wayne@usa.com	22222222	Texas

### **Django Files and Folders**



### 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.

### Create the Model (Table)

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



### "models.py"

from django.db import models

```
# Create your models here.
class Customer(models.Model):
    first name = models.CharField(max length=30)
    last name = models.CharField(max length=30)
    email = models.EmailField()
    phone = models.CharField(max length=15)
    address = models.TextField()
    def __str__(self):
```

```
return f"{self.first_name} {self.last_name}"
```

### Create the Template

Create a new folder called "**templates**". Then create a new Template, i.e., a HTML File called "**customers.html**" in the /templates/ folder:

File Edit Selection View Go Run Terminal Help Customers.html COMPANY customers > templates > <> customers.html > <> body > <> div.container-fluid.pt-5 > <> div.table-responsive > <> table.table > <> tr > <> td company 2 <html> > \_\_pycache\_\_ \_\_init\_\_.py <body> 10 asgi.py settings.pv <div class="container-fluid pt-5"> 11 urls.py <div class="table-responsive"> wsgi.py 14 ✓ customers > \_\_pycache\_ 15 v migrations Lastname > \_\_pycache 18 \_init\_.py Email 0001 initial.pv 19 v templates Phone customers.html 20 init .pv Address 21 admin.py apps.pv 22 models.py tests.py 23 {% for customer in customers %} 🔹 urls.py views.py > 24 db.salite3 manage.py 25 {{ customer.first name }} {{ customer.last name }} 26 {{ customer.email }} 27 {{ customer.phone }} 28 {{ customer.address }} 29 30 {% endfor %} 31 32 </body> 33

Templates in Django are basically just HTML files.

### "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>
                                                                           Note! Bootstrap is used to
  Here you see the customers in the database:
  <div class="table-responsive">
                                                                            boost up the appearance
  >
                                                                           of the web page.
         Firstname
         Lastname
         Email
         Phone
         1+hr Addmacc / /+hr
      {% for customer in customers %}
                                             Django code that retrieves data from
         >
            {{ customer.first name }}
                                             the "Customer" Model, which again
            {{ customer.last name }}
            {{ customer.email }}
                                             retrieves data from the customer
            {{ customer.phone }}
            {{ customer.address }}
                                             table in the PostgreSQL database.
         {% endfor %}
  </body>
</html>
```

### Create the View

⋞	File Edit Selection View Go Run	Terminal	$\leftarrow \rightarrow$	,	<b>8</b> ~					
ŋ	EXPLORER	models.py	customers.html	🕏 views.py 🛛 🗙		$\triangleright$ $\checkmark$ $\square$ $\cdots$				
-		customers > 🔹 views.py > 😚 customers								
ρ	✓ company >pycache	1	from dja	ngo.http <mark>import</mark> HttpF	Response					
ц	<ul> <li>initpy</li> <li>asgi.py</li> </ul>	2	from dja	ingo.template import ]	loader					
å	🔹 settings.py 🔹 urls.py	3	from .mo	dels import Customer						
₿	<ul> <li>wsgi.py</li> <li>customers</li> </ul>	4								
Д	>pycache ~ migrations	5	# Create	e your views here.						
	>pycache initpy	6	def cust	<pre>comers(request):</pre>						
	<ul> <li>0001_initial.py</li> <li>v templates</li> </ul>	7	cust	comers = Customer.obje	ects.all()					
	customers.html	8	temp	<pre>late = loader.get_ten</pre>	nplate <mark>('custome</mark> r	rs.html')				
	<ul> <li>admin.py</li> <li>apps.py</li> </ul>	9	cont	:ext = {						
	<ul> <li>dppspy</li> <li>models.py</li> <li>tests py</li> </ul>	10		'customers': customer	^s,					
	<ul> <li>views.py</li> <li>db solite3</li> </ul>	11	}							
8	<ul> <li>manage.py</li> </ul>	12	retu	<pre>Irn HttpResponse(temp]</pre>	late.render(cont	<pre>:ext, request))</pre>				
503	> OUTLINE									
205	> TIMELINE									
~	$\otimes$ 0 $\wedge$ 0				Ln 9. Col 16 Spaces: 4 UTF-8 CRLF	() Python 👸 3.12.6 ('diangoeny': yeny)				

### "view.py"

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

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

### "urls.py"

Create a "urls.py" file for the "customers" Django App:



7

### **Customer App**

Then run your Django Project: ...>python manage.py runserver

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



### Add some Data

#### We can use, e.g., pgAdmin to insert some data into the table:

customers_customer		
🗸 📋 Columns (6)		
id 🔋		
🚦 first_name		
🚦 last_name	🚏 pgAdmin 4 – 🛛	×
email	File Object Tools Edit View Window Help  Welcome  Company/postgres@PostgreSQL 17* ×	
phone	S     Company/postgres@PostgreSQL 17     V       S     D     D       V     Notimit     D       D     D     V	٥
address	Query Query History	~
	insert into customers_customer(first_name, last_name, email, phone, address) values ('Elvis', 'Presley', 'elvis.presley@usa.com', '11111111', 'Memphis')	
	Data Output Messages Notifications	2
	INSERT 0 1	
	Query returned successfully in 39 msec.	
	Total rows: Query complete 00:00:00.039	37

### **Customer App**

We now get data that we inserted in the PostgreSQL database (using "pgAdmin") in our Customers App:

	<ul> <li>✓ ③ Customers</li> <li>← → ○ ④ 12</li> </ul>	× + 7.0.0.1:8000/customers/			_ ☆	□ ②	×
	Customer Here you see the custo	<b>S</b> mers in the database:					
	Firstname	Lastname	Email	Phone	Address		
	Elvis	Presley	elvis.presley@usa.com	1111111	Memphis		
URL: <b>127.0.0</b> .	1:8000/cust	omers					

#### https://www.halvorsen.blog



# Django Admin

#### Hans-Petter Halvorsen



### Django Admin

#### http://127.0.0.1:8000/admin

👻 🔇 Log in   D	Django site admin × +	_		×
$\leftarrow \rightarrow $ C	① 127.0.0.1:8000/admin/login/?next=/admin/	☆	0	:
	Django administration <b>O</b>			
	Username:			
	Password:			
	LOG IN			

"Django Admin" interface is included with Django.

### Django Admin

- "Django Admin" is included with Django.
- "Django Admin" offers a CRUD user interface for all your Models.
- CRUD is short for Create, Read, Update and Delete.
- Note! The Admin Interface is typically intended for "superusers" and no for ordinary users of your application.
- To use it you need to create a User.

### Django Admin - Create User

Type the following to create a new User:

...>python manage.py createsuperuser

 (djangoenv) PS C:\Temp\Development\djangoenv\company> python manage.py createsuperuser Username (leave blank to use 'hansp'): Email address: hans.p.halvorsen@usn.no Password: Password (again): This password is too short. It must contain at least 8 characters. Bypass password validation and create user anyway? [y/N]: y Superuser created successfully.
 (djangoenv) PS C:\Temp\Development\djangoenv\company> []

Then run the server:

...>python manage.py runserver

Goto <a href="http://127.0.0.1:8000/admin">http://127.0.0.1:8000/admin</a> and enter your Username and Password:

Then you are asked to create a Username and a Password.

Django administration **O** 

Password:

### Django Admin

#### The following will appear:



Here you can Add, Change and Delete (CRUD functionality) **Groups** and **Users**.

How can we add Customers?

### **Customer App**

We now get data that we inserted in the PostgreSQL database (using "pgAdmin") in our Customers App:

	<ul> <li>✓ ③ Customers</li> <li>← → ○ ④ 12</li> </ul>	× + 7.0.0.1:8000/customers/			_ ☆	□ ②	×
	Customer Here you see the custo	<b>S</b> mers in the database:					
	Firstname	Lastname	Email	Phone	Address		
	Elvis	Presley	elvis.presley@usa.com	1111111	Memphis		
URL: <b>127.0.0</b> .	1:8000/cust	omers					

### Add Customer Model

Django Admin offers a CRUD user interface for all your Models. Let's add CRUD functionality for our Customer Model.

You need to register the Customer Model in the file "admin.py":



Then go back to http://127.0.0.1:8000/admin



### **Customer CRUD Interface**

•	S Customer	s	× €	Select customer to change   Dja × +			_		×					
←	$\rightarrow$ C	① 127.0.0.	.1:8000/admin/o	customers/customer/			☆	2	:					
	Django a Home > Custo	administ	ration		<ul> <li>✓ S Customers</li> <li>← → C O 127.0.</li> </ul>	× S Eh	ris Presley   Change customer ×	+				_ ☆	•	× :
Sta A	art typing to fil AUTHENTICATI Groups	lter ON AND AUTH	HORIZATION + Add	Select customer to change	Django adminis Home > Customers > Cust Start typing to filter AUTHENTICATION AND AU	stration tomers > Elvis Presley UTHORIZATION	Change customer				WELCOME, HANSP. <u>VIEW SITE</u> / CHANGE PAS	ISWORD / LOG OL		
« «	Jsers CUSTOMERS Customers		+ Add + Add	customer     customer	Groups Users CUSTOMERS Customers	+ Add + Add + Add	First name: Last name: Email:	Elvis Presley elvis.presl	ley@usa.co	m				
					~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		Phone: Address:	11111111 Memphis	1					
Now you can Create, Read, Update and Delete (CRUD functionality) Customers.							SAVE Save and ad	id another	Save a	nd continue editing	9	Dele	te	

### **Customer App**

Here is our final Customer App where the end-user see a list of available Customers, while a superuser or administrator can Create, Read, Update and Delete Customers in the PostgreSQL database and presented in the Customers web page.

~	<b>O</b> (	Custome	s				×	Select customer to change   Dje × +	-		×
←	$\rightarrow$	C	0	12	7.0.0.	1:800	0/cu	tomers/	☆	2	÷
Customers											

Here you see the customers in the database:

Firstname	Lastname	Email	Phone	Address
Elvis	Presley	elvis.presley@usa.com	11111111	Memphis
John	Wayne	john.wayne@usa.com	22222222	Texas

### Summary

- We created a Django Project and a Django Web Application.
- 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.
- In this Tutorial we used **Django** in combination with **PostgreSQL**.
- We created a simple "**Customer App**" that retrieves a list of Customers stored in the PostgreSQL database.
- Finally, we used the built-in "**Django Admin**" module to create **CRUD** functionality, i.e., the possibility to Create, Read, Update and Delete Customers in the PostgreSQL database.

### Hans-Petter Halvorsen

**University of South-Eastern Norway** 

www.usn.no

E-mail: <u>hans.p.halvorsen@usn.no</u> Web: <u>https://www.halvorsen.blog</u>

