Digital Resume

Using Python backend & Django



By Sunil Dangal

Table of  
Contents

LET’S Begin… Introduction 1

Template 1

Backend 2

Frontend 6

# LET’S Begin… Introduction

In this Project, I am going to create a resume website and digitize it using Django and Python. The reason I am using Django is because of two options. Option 1 is, it has a built-in admin page which allows us to create read, update, and delete objects which will avoid the need to update the source code in this project. Therefore, we can add objects in the admin page, and it gets rendered straight into the html. Option 2, it gives us opportunity to refine Django skills.

Firstly, we are going to:

1. Select template.
2. Django Project.
3. Build backend.
4. Build frontend.
5. String together

## Template

So, we have a template on a GitHub provided by Bobby Stearman the developer guy. We are going to clone the template to our repository or in our machine and string that template into a Django project.

So, in the template, we have **Home, Portfolio, Blog, and Contact** pages. All the pages will be rendered to the front-end with a rich text editor and we will also add to Django admin. Whatever we add in a rich text editor will be rendered directly as it’s seen in admin to the front-end. We will be creating objects in Django admin without needing to manipulate the source code for this project.

Let’s Start:

# Backend

So, before starting Django project, we must create a virtual environment. we must open IDLE, and in terminal we are going to install Django, Pillow and Django ckeditor *[pip install django Pillow django-ckeditor]*. You might also need to upgrade pip if the upgrade pip pops up *[pip install --upgrade]*. Once all these are installed, we will then start Django project. *[django-admin startproject digital\_resume]* Now we can *[cd digital\_resume].* Now the virtual environment is started, and we can do *[python manage.py] startapp main* which will create main folder.

This project will require the use of static files. The reason we have [import os] is because we are going to use that later to join the base directory to the new static file directories. We also need to add main, ckeditor in the [INSTALLED\_APPS].

There is a MIDDLEWARE which we don’t need, and we are going to add a context processor.

We would then add path’s in [urls.py] and bring on the Django’s settings. We also need static from [conf.urls static]. then we will add a path for [main.urls] and all the urls will be held inside urlpatterns.

In [if settings.DEBUG], we are going to add static URL and static route to the URL. Like for [MEDIA\_URL] and [MEDIA\_ROOT]. Therefore, we will easily have static directory and media files directory separately.

Now we are going to create another python file inside [main] which is [urls.py]. Inside it, we are then going to reference view inside main i.e., [urls.py] with [urls.py] in digital\_resume. We are also going to make some migrations.

Let’s start with models.py, which translates into database tables. We are going to create a model and the names becomes the table name and fields becomes the columns and then each object that we save becomes each row in that table ( Django’s Model).

Now inside, models.py, we are importing models, there is a built-in user model in [contrib.auth] and models. We can extend that for user profile. We are going to use signals to extend it. We then import sluggify because we need slug for our blog and our portfolio. And at last, we are importing django ck editor(rich text field). We can add this rtf in our blog and portfolio.

Inside, we are going to create a skill model. Inside the skill models we have Key skills and coding skills where it is represented in a bar with percentage. Likewise, Images used in each Key skill are svg format images are used. And then in class User Profile, it extends the built-in user model and in this case of One-to-One field here on delete cascade which produce one user, we are the super user. We have an avatar which is an image field.

In class Media(models.Model): If we can add an image to the media model then that image gets saved into the media directory within media file directory. We can then have access to that url when we are creating the rich text editor image. It just a simple way for accessing the stacked files in a template when we are messing around in a admin.

Now in admin.py we will register all the models to access them. Inside this, there is [readonly\_fields = (‘slug’) and we don’t want to be changing this because a slug could be used i.e., you could send a blog link and it could have a back link somewhere for a blog, so we don’t want to be changing that slug then the link becomes inactive.

Now we then need a way for creating user profile using signals. So, we create another python file inside main which is [signals.py]. We are getting the user profiles which we created inside signal. Signal is like a receiver because when a user object is created then it gives the signal to this signal.py therefore @receiver picks up this signal and the function create\_ profile contains the arguments(sender), so therefore if the user object is created then we want to create user profile.

We also need to wire the signals.py to the apps.py by overriding using ready method.

Now, we are going to create a models form inside the main. Inside it, we don’t need the classes, so we delete them because, we are not using form control in the template.

Now we need views, and we got the imports inside it. Inside import messages denotes that when a form is valid and is saved, then it will render the message pops up “thank you for submitting the form….”. We are also creating a directory in main, i.e., templates and inside templates we need another called as main. Inside main where we add our templates.

In views.py > context indices like ‘testimonials’ we can reference them which contains the list of objects that we can render.

Here in the console, we have not created **Requirements** files. So now write *pip freeze > requirements.txt* which will create the requirements.txt file and inside it contains the requirements for the project. Ckeditor is the rich text field editor, pillow for controlling images.

Now we are going to make migrations:

In terminal 1st write *python manage.py makemigrations*

And write *python manage.py migrate*

So, after running above commands: I got error which is:

WARNINGS:

: (staticfiles.W004) The directory '/Users/sunildangal/Desktop/IT Projects/Resume/digital\_resume/media' in the STATICFILES\_DIRS setting does not exist.

: (staticfiles.W004) The directory '/Users/sunildangal/Desktop/IT Projects/Resume/digital\_resume/static' in the STATICFILES\_DIRS setting does not exist.

What I did is I made the folder media and static inside the directory.

Now again, in terminal type:

* python manage.py createsuperuser # this will allow us to create superuser

So, I used my email and created a password: \_\_\_\_\_\_\_\_\_ and username.

Now the super user is created.

So, lets type *python manage.py runserver* Note: We wired the main app into our url file so, everything that appears inside urls.py are specific to main. Therefore, the first path is blank which is going to be our home page i.e., our localhost port 8000 with index.as view because we are using class-based views so, we must use dot. In other way, if it was function-based view we didn’t needed to use dot. So, that the way to reference it in the template.

So, let’s go to the terminal and follow the link. When we open that, we can see that we don’t have the templates.

Let’s go to admin page on 127.0.0.1:8000/admin and sign in with the credentials that is added before and then we can see the Django administration page.

So, we can see authentication model inside users, and we can see sunildangal account that we created. We can put our first name and last name. (Could be done from terminal too)

So that signal should have generated user profiles which is ID’d 1 in user profiles. In here, we can choose avatar and do all sort of things. Basic We are building on the admin page which is the perks of using Django. The build-in admin page allows us to create easily and efficiently so that anytime we can update details like avatar, bio, cv, etc. For examples: in a production environment, we don’t have to go to the source code, instead, we can just change the back end.

In Django administration page > inside Testimonials we can add testimonials and make active and in in-active state (won’t be rendered in front-end if it’s deactivated).

We can add Media files, certificates, and Blog profile. For add in Blog Profiles, we need rich text editor.

Notice that when we open different ID from different fields, we have different url. For example: This is the url of the image inside media files,

<http://127.0.0.1:8000/media/media/django_walkthrough_5_UqLJMWx.png> (you get this link when you open image inside Media Files)

where we have saved into our static files. So, if we do all these, basically this is the backend done.

Let’s Start:

# Frontend

Now we are going to wire in the front-end. So, we have got the templates inside the main. What we need now are static files. The files that don’t change much like css, images and jss. So, we are going to copy css, images and jss folder and paste it inside the static folder inside digital\_resume>main>templates>main. Now, we need to wire them (html files).

Let’s open the index.html in IDLE and then let’s render it (run). It has found the index.php, the template, but the full functionality is not shown yet because its referencing files that django doesn’t know where to find them. So, let’s go to the terminal and type:

*workon digital\_resume* or *cd digital\_resume*

If the workon …. Command doesn’t work i.e., if it shows error zsh: command not found: workon then you must install virtualenvwrapper first.

<https://virtualenvwrapper.readthedocs.io/en/latest/install.html#shell-startup-file>

and if the workon works then skip this step and stary typing:

* python manage.py runserver

So. after opening the link, we can see the page doesn’t look perfect, there is no css, jss, images and all the contents. Therefore, we are going to load the static files.

SO, we are going to save as index.html into base.html in the same directory and then we are going to create a new directory inside the templates/main/partials. Inside that new directory partials, we are going to create 3 HTML file which are: footer.html, nav.html and messages.html.

We can then dissect up the base.html, and other html’s and use template tags. Django’s have heaps of template tags that allows us to use other parts of html, and all we need to do is reference the path.

At the top of the page base.html, write:

{% load static %}

This is a template tags that allows us to access the static files.

Inside base.html, in canonical we need to add context processor like in settings.py, we have added context processor before.

So, we are extending the main/base. Therefore, when we render the homepage, it’s looking for index.html and it’s going to put all the information in that html in the blocks we have stated.

So, everything we have put in index.html, those will be rendered in the block there in base.html.

Inside idex.html, we are going to replace “images/ with “{% static ‘images/ everywhere. In PyCharm, select the code and press command + R for replacement. Likewise, change .jpg” to .jpg’ %}” And, change .svg” to .svg’ %}” and replace all. So, that’s all we need to do for this index.html page. So, we must do exactly same for all the pages like we did with index.html page.

So, we have now got a digital resume. If you there is issue understanding this documentation, please check the link:

<https://www.youtube.com/watch?v=0oSsLbh_Kv4>

and follow up if you want to make your resume digital too.

Thanks to Bobby Stearman guy from YouTube freeCodeCamp.org and thanks to Bobby’s friend, James Granger Design for letting us use his template for this project.

Also likewise, now what we are going to do is host this resume in AWS cloud and run it from here.

**Note:** Some photos were created using Microsoft Bing Image creator. So, shoutout to Microsoft AI for the photo.

Special Thanks to Bobby Stearman for his tutorial on youtube and James Granger for his HTML template.