Profile Form:

What it accomplishes?

The profile page (/blog/profile denotes profile settings) provides individualized information, based on the account that is currently logged in. This page provides the profile picture, ID, username, email, first name, and last name associated with an account. Furthermore, it provides the option to alter the content of these unique fields (e.g, being able to change the profile picture and/or changing the email associated with the account). The only field that cannot be altered is ID. Upon a successful update of account information, a message prompts within the profile page that the information has been successfully updated.

How does this technology accomplish what it does?

The profile form is located within 'blog/templates/blog/Profile.html', and inherits the template file located at 'blog/base.html'. This page contains the skeleton code that consists of the general structure of the html and css of Profile.html. Django's 'Crispy Forms' and bootstrap (bootstrap 4) is utilized to provide a better aesthetic to the form, namely through well structured tables and classes. Aside from the style, note that Profile.html also utilizes the Django templates ability to load variables (e.g, first name, username, email).

When a user/member is logged in, and clicks on the profile tab, a request is sent and received by the profile() function in /blog/views.py. What this method does is utilize both UserUpdateForm() and PictureUpdateForm() located in /authentication/forms.py. Both of these functions utilize django.contrib.auth.models (namely the class 'User' linked here), forms (namely the class 'ModelForm' linked here), and django.db (namely the class 'models' linked here). When a request is received, logic is split by either the 'POST' or 'GET' method:

1. If it is simply a GET request, the profile page displays the Profile.html template, along with data retrieved from both the UserUpdateForm and PictureUpdateForm. With regards to the UserUpdateForm, it utilizes the 'User' model to create a form containing all current values for username, email, first name, and last name of the logged in user/member. Furthermore, the profile() function utilizes the PictureUpdateForm, which uses the 'Profile' model to grab values of the user and (more importantly) the image associated with the user/member. Both of these models ('User' and 'Profile') utilizes django.db (class 'models'), which contains our database of user/member information. Upon being able to grab information about the user/member (both personal account information and profile picture), this is stored within info_form, and picture_form respectively. Utilizing django.shortcuts (namely render() linked here), Profile.html is able to render this data through the context of info_form and picture_form (loads these variables)
2. Upon clicking on the update button on the profile page, a POST request utilizes the JSON format to extract the information within each respective field (e.g, username, email, or image). UserUpdateForm() and PictureUpdateForm() are utilized to attempt to update the corresponding forms. If there are no issues with this process, the database information associated with a logged in member is updated/saved with the new information (utilizes save() linked here). Upon being able to grab information about the user/member (both personal account information and profile picture), this is stored within info_form, and picture_form respectively. Utilizing django.shortcuts (linked here),

Profile.html is able to render this data through the context of info_form and picture_form (loads these variables).

What license(s) or terms of service apply to this technology?

The entire application runs on the Django framework. The owner of the license of all Django libraries is Django Software Foundation as mentioned here. They give the right to use Django under many circumstances mentioned in the link provided.

It is mentioned that you can use the libraries within the following conditions:

1) Service Identification
2) Django-related software project
3) Groups and Events
4) Merchandise
5) Products and services serving the community
6) Other Commercial activity
7) Domain Names
8) Uses outside of this license
9) Community standards
10) Interpretation

django-crispy-forms:

What it accomplishes?

Django-crispy-forms works to control the structure/style of Django forms, without the hassle of having to write individual custom form designs. For example, django-crispy-forms can be used to align text fields that may be grouped together (one on top of the other). This allows for error messages to be less confusing with respect to which text field creates the error.

How does this technology accomplish what it does? (how it's implemented linked here)

Within any html file or Django template, the end user must import django-crispy-forms with {% load crispy_forms_tags %}, and invoke the keyword "|crispy" with respect to a context (e.g, the textfields of a form). django-crispy-forms then adjusts form properties in the backend, utilizing bootstrap (bootstrap 4) template pack to automatically create html code for each form field. It is known that bootstrap is the template pack, as the variable CRISPY_TEMPLATE_PACK in class_project/settings.py is set to bootstrap4.

What license(s) or terms of service apply to this technology?

This useful library has code that is hosted on Github, with a MIT License (linked here). Based on

this license, this technology can be used commercially. It can also be modified, distributed, and used for private purposes. The contributors to the project are not liable for any damage. There is no warranty for the library.