Samuel Kiragu

6 Followers

Using Curl To Test Django Views

Samuel Kiragu 1 day ago · 4 min read

I had just finished writing my Django view(a function or class that receives requests, handles the request, and returns a response). However, I had not linked it to its relevant GUI(Graphical User Interface) because it was non-existent. Did this mean I couldn't test my views functionality? I had to find a way, a way to test my views without having to have a fully developed GUI.

Using Curl, I tested my Django application's view for handling user registration. If you've been in a situation like mine, or are curious about how Curl can be used to test a Django application, this is the article for you. Ready? Read on!

What is Curl

Curl is a command-line tool used to transfer data using URLs. It has been in existence since 1998. Since then, the tool has been used daily by virtually every Internet using human on the globe. Surprising, right?

Curl is the Internet transfer engine for thousands of software applications in over ten billion installations

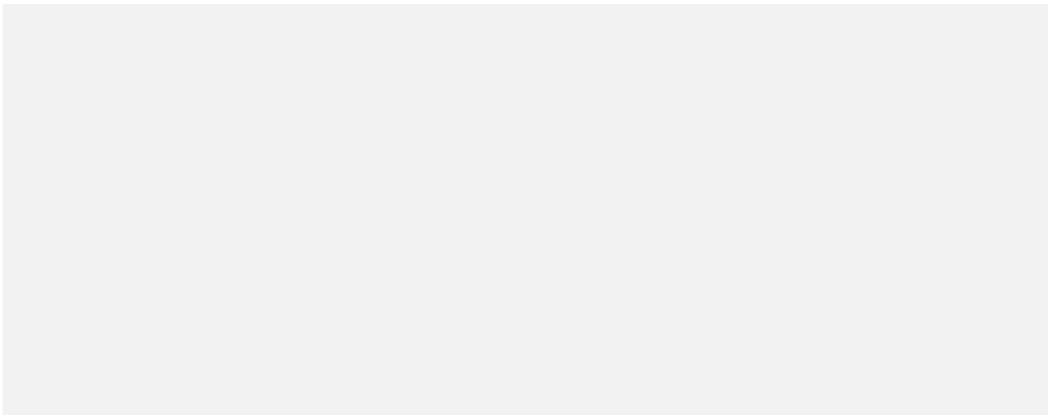
Configuring Django Project For Testing

To follow through with these steps, you'll need a Django application. You can use the Django application of your choice. You can use this tutorial if you haven't created a Django project yet. To follow through these are the configurations you ought to make

1. Navigate to your Django project

I used my terminal to navigate to my Django project. Alternatively, you can use the File Manager provided by your Operating System to navigate to your Django project.

In my case, the project is called a project.



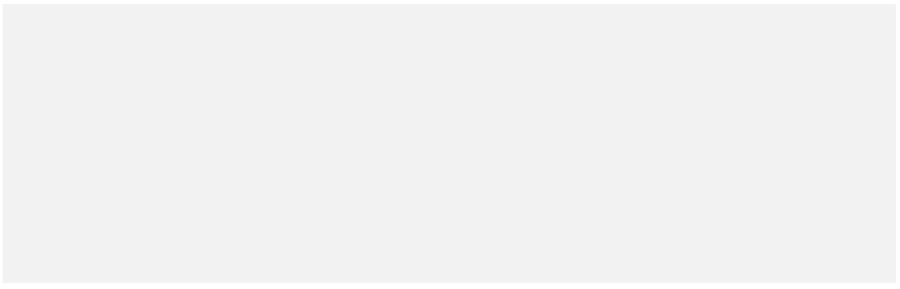
Directory Containing my Django Project

2. Open the settings.py module.

The settings.py module is located inside the project's directory. Using the editor of your choice opens the module.

3. Comment django.middleware.csrf.CsrfViewMiddleware in the MIDDLEWARE list.

django.middleware.csrf.CsrfViewMiddleware offers your web application a layer of security against CSRF(Cross-Site Reference Forgery) attacks by adding a CSRF token to a web page. By enabling this middleware, a CSRF token will be appended to each posted form. This will however add some complexity when testing the form's functionality using curl. For testing purposes, I commented on the middleware. However, make sure you enable this middleware when you deploy your application.



Commenting Your CSRF Middleware

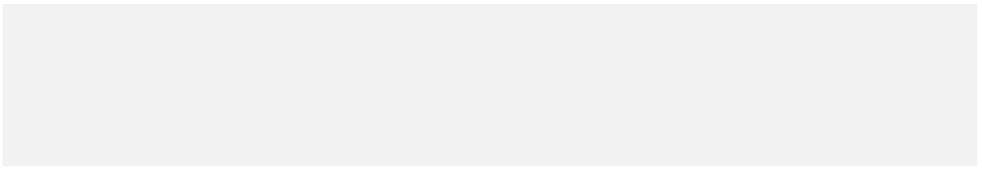
4. Write a view function or class to handle form response.

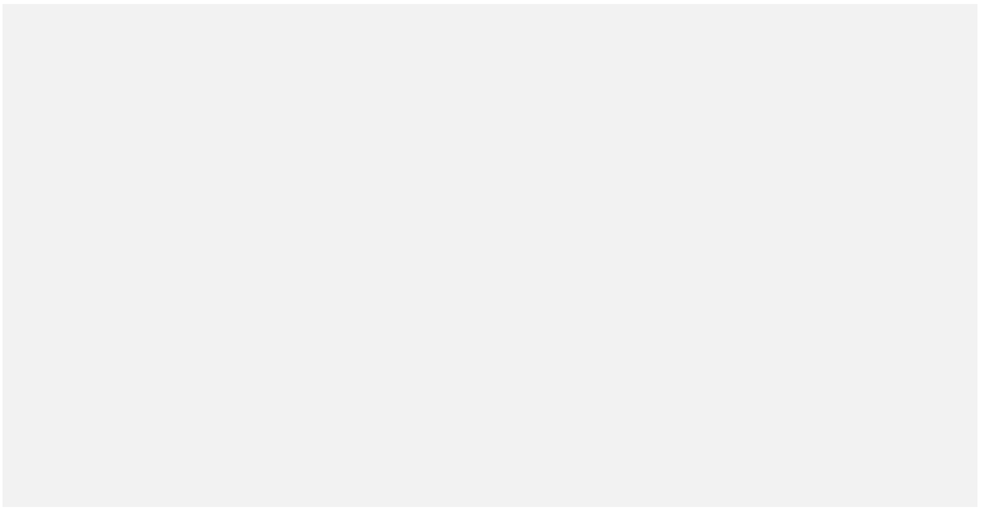
I used a function to handle my form data.

Inside the function, I checked the type of request and whether each request contained an email and password.

Only requests of type **POST** were accepted. If a request contained both a password and an email, they would be saved to the database using **the UserProfile** class, a Django model. However, the password would first be encrypted before storage.

If a request wasn't of type POST or did not contain both an email and password fields, it would return "Failure". The response is also returned if you try to register using an already existing email.





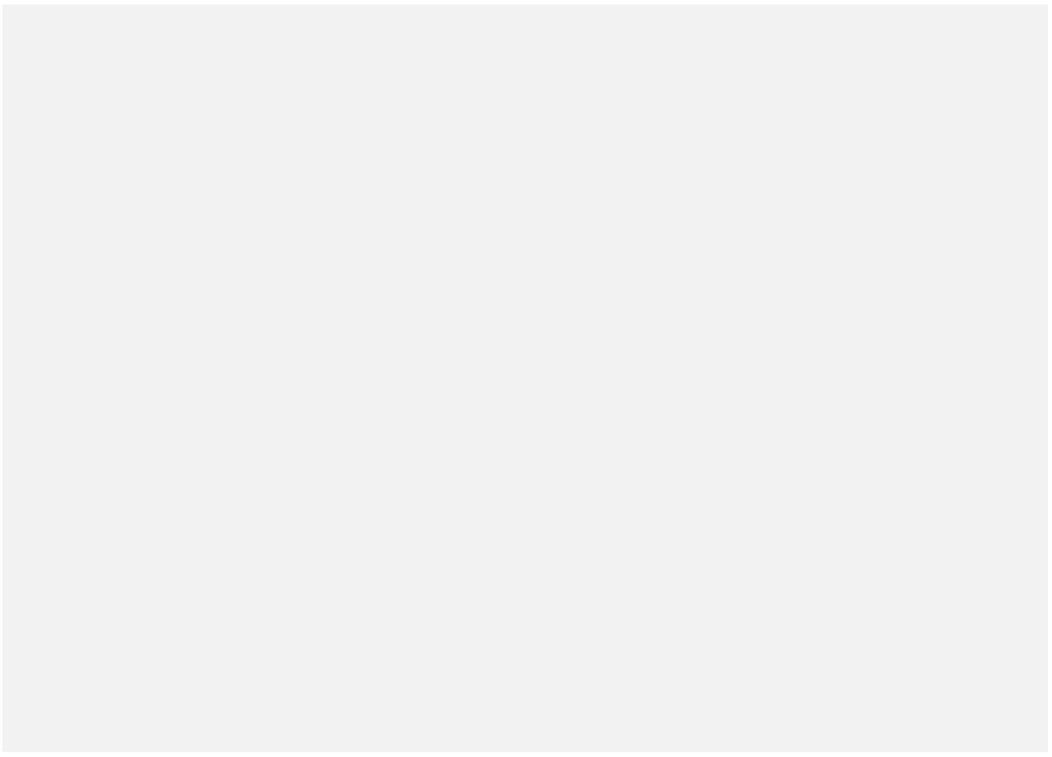
View handling form registration data

Remember to create a path that links your view with a relevant url_pattern in urls.py

Testing The View Using Curl

Understanding the command

Great! Your Django application is ready for testing. Let’s now break down the curl command and understand what is happening.



— data-urlencode: This is an option made available by curl that performs URL Encoding. I used it because the email contains curl’s reserved character “@”, used to specify files.

I specified the field name and its value and enclosed them in quotes. However, feel free to replace them with field names and values relevant to your site.

Finally, the last argument in the command above is the URL. It is a required argument when using the curl command.

Executing the command

After running the command the first time, the HTTP Response was “Success”. This was because the account specified was none existent. After running the command the second time, the HTTP Response was “Failure”. This was because the account had already been registered.

Conclusion

Without interacting with the browser, we’ve been able to test that the register view works just fine. We’ve done this using curl, a tool used to transfer data using URLs. This was the use case I found for the tool. Explore and find others.

Curl Django Python Testing