**How to render data in Django**

https://www.educative.io/answers/how-to-render-data-in-django

render(request, template, {})

In Django, **render()** is one of the most used functions that combines a template with a context dictionary and returns an HttpResponse object with the rendered text.

### Arguments

It takes the following three arguments.

1. **request**: This is everything we receive from the user via the Internet.
2. **template**: This is the template file that will be rendered.
3. **context dictionary**: This is a dictionary where we store data that can be used by the template.
4. from django.shortcuts import render  
     
   def about(request):  
       template = "about.html"  
       context = {  
         fullName: "Shubham Kshariya",  
         age: 22,  
         destination: "Software Engineer"  
       }  
       return render(request, template , context)

### Explanation

First, we need to import the render library from django.shortcuts.

After that, we create an about function where we define a template file and a context dictionary.

At the end of the function, we will invoke the render() function and pass the required parameters to it.

### Rendering data

The render() function will return an HttpResponse including the about.html and context dictionary.

In order to render a dynamic variable in Django, it needs to be wrapped between {{ and }}.

Check out how to create the about.html to render data:

<html>  
  <head>  
    <title>User Information</title>  
  </head>  
  <body>  
    <p>Name: {{ fullName }}</p>  
    <p>Age: {{ age }}</p>  
    <p>Designation: {{ designation }}</p>  
  </body>  
</html>

### Explanation

The dynamic portion of the file is wrapped between {{ and }}.

We can also implement dictionary lookup, attribute lookup, and list-index lookups using a dot notation.

### Adding comments

A comment can be also be added in the template like this.

{# single line comment #}  
  
{% multiple line comment %}

# Request and response objects

Django uses request and response objects to pass state through the system.

When a page is requested, Django creates an **[HttpRequest](https://docs.djangoproject.com/en/5.0/ref/request-response/" \l "django.http.HttpRequest" \o "django.http.HttpRequest)** object that contains metadata about the request. Then Django loads the appropriate view, passing the **[HttpRequest](https://docs.djangoproject.com/en/5.0/ref/request-response/" \l "django.http.HttpRequest" \o "django.http.HttpRequest)** as the first argument to the view function. Each view is responsible for returning an **[HttpResponse](https://docs.djangoproject.com/en/5.0/ref/request-response/" \l "django.http.HttpResponse" \o "django.http.HttpResponse)** object.

This document explains the APIs for **[HttpRequest](https://docs.djangoproject.com/en/5.0/ref/request-response/" \l "django.http.HttpRequest" \o "django.http.HttpRequest)** and **[HttpResponse](https://docs.djangoproject.com/en/5.0/ref/request-response/" \l "django.http.HttpResponse" \o "django.http.HttpResponse)** objects, which are defined in the **[django.http](https://docs.djangoproject.com/en/5.0/ref/request-response/" \l "module-django.http" \o "django.http: Classes dealing with HTTP requests and responses.)** module.

+++++++++++++++++++++++++++++++++++++++++++++++++++

**What are HTTP Request Methods:get and post methods**

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client.

GET is used to request data from a specified resource.

Note that the query string (name/value pairs) is sent in the URL of a GET request:

/test/demo\_form.php?name1=value1&name2=value2

-----------++++--------------

GET requests can be cached

GET requests remain in the browser history

GET requests can be bookmarked

GET requests should never be used when dealing with sensitive data

POST is used to send data to a server to create/update a resource.

The data sent to the server with POST is stored in the request body of the HTTP request:

POST /test/demo\_form.php HTTP/1.1

Host: w3schools.com

----------++-------

POST requests are never cached

POST requests do not remain in the browser history

POST requests cannot be bookmarked

**Create form using post method with CSRF in Django.**

By using safe and secure HTTP, your data also stay secure. The post method goes through HTTP headers, which are known as QUERY\_STRING in Django. If you have applied CSRF token form cookies, you can stop the hackers and save your website from potential threats. Django will check the information if you are using CSRF in the post method.