**About django installation and all command required during programm.**

Python

Pip install django

Or

Python -m pip install django

Python -m django --version

Python -m django

Python -m startproject urprojectname

PS C:\Users\piyus\PycharmProjects\textutils\textutils> python manage.py runserver

To make changes-python manage.py makemigrations

To save the changes-python manage.py migrate

**Understanding working as well as coding of project in front end and back end**

**1. 05:44 - Creating views.py :**

Now, I am going to create a file called views.py inside the textutils folder (The folder which was created for us by Django)

In this file, we will define views that will perform some specific functions for our website

**2. Building urls.py :**

Now, we will see the working of URLs in Django. First of all, we need to import views in our urls.py file. Open urls.py file and type the following command :

from . import views

The single dot is a convention for the current directory. We have already created views.py file in our previous tutorial. If you have any doubt read the previous tutorial here.

Now, type the following code in urls.py file :

urlpatterns = [

path('admin/', admin.site.urls),

**5.What is the path function?**

**Path function is contained within the django.urls module.**

**It helps in routing URLs to the appropriate view function within a Django project.**

Now, let' s discuss the arguments passed in the path function.

1. about/: It is the endpoint of the URL.

Example: Let's suppose a user tries to access https://www.codewithharry.com/blog/. In this URL, 'blog' is the endpoint.

2. views.about: It is the function defined in the views.py file. Here, we are passing the function that should be executed whenever someone tries to access the 'about' page of our website.

3. name='about': It is the name of the path. Naming a path will help us to access it from anywhere in our project. In future, we will see how to access a path from templates.

**6.**

**3:35 - Creating Functions In views.py :**

Open views.py file and write the following code:

from django.http import HttpResponse

def index(request):

return HttpResponse('''Harry Django CodeWithHarry''')

def about(request):

return HttpResponse("About Harry Bhai")

Now, let's understand the above code line by line:

**First of all, we have imported the HttpResponse object from django.http module.**

**Whenever a user requests a page, Django creates an HttpRequest object that contains the metadata of the request. After creating an HttpRequest object, Django passes this object inside the view function as the first argument. Hence, a specific HttpResponse is generated for each view function**.

In the second line, we have created a function named index, which returns "Harry Django CodeWithHarry" as HttpResponse. Each view function takes request as a default argument. Similarly, we have created another function called about which returns "About Harry Bhai."

**8.**

**Django Website: Laying The Pipeline** | Python Django Tutorials In Hindi #7

In the tutorial, we will build a pipeline for our textutils website.

00:00 - Introduction :

Laying the pipeline of a website is like creating a blueprint for the website. In this tutorial, we will build various endpoints that will help us to get a rough idea about the working of the website.

**9**

00:43- Creating Home Page Of TextUtils Website :

A homepage is the first thing people see when they visit any website. Homepages are like digital welcome signs to your user. So, without wasting time, let's start building the homepage of our textutils website.

Step 1: Open views.py file and type the below code:

from django.http import HttpResponse

def index(request):

return HttpResponse("Home")

We have already seen the working of the index function and HttpResponse module in our previous tutorials. If you have any doubt, then click here and follow the tutorials.

**10.**

02:47 - Creating removepunc Endpoint :

Type the following code in views.py file :

def removepunc(request):

return HttpResponse("remove punc")

In the above code, we have created a function named removepunc, which returns "remove punc" as a HttpResponse. After defining a function in views.py, it is essential to map the view function inside urls.py file. Otherwise, you will receive "Page Not Found" error on accessing the end-point.

Open urls.py file and type the following code to map the removepunc function:

path('removepunc', views.removepunc, name='rempun'),

03:55 - Creating Capfirst Endpoint:

Now, we are going to create our second endpoint named capfirst. Write the following code in views.py file:

def capfirst(request):

return HttpResponse("capitalize first")

As we have successfully created the capfirst function, let's map this function in urls.py. Open urls.py and type the code given below:

path('capitalizefirst', views.capfirst, name='capfirst'),

04:50- Creating Some Other endpoints:

Now, we are familiar with the process of creating a new endpoint from scratch. Let's create some more endpoints for our website.

newlineremove: Let us call the endpoint as newlineremover. It will have the following url and view function associated with it:

Function code:

def newlineremove(request):

return HttpResponse("capitalize first")

URL mapping:

path('newlineremove', views.newlineremove, name='newlineremove')

spaceremove: Let us call the endpoint as spaceremove. It will have the following url and view function associated with it:

Function code:

def spaceremove(request):

return HttpResponse("space remover")

URL mapping:

path('spaceremove', views.spaceremove, name='spaceremove')

charcount: Let us call the endpoint as charcount. It will have the following URL and view function associated with it:

Function code:

def charcount(request):

return HttpResponse("charcount ")

URL mapping:

path('charcount', views.charcount, name='charcount')

And, that's how you can easily create a simple yet efficient pipeline for any website. Follow the steps carefully, and do not forget to check whether all the end-points are working correctly or not. If you receive any error, feel free to ask in the QnA section.

**11.**

**What Is A Django Template?**

**It is a .html file that contains HTML, CSS and Javascript.**

**Django templates are written in the template language. Don't worry! You don't need to learn a new language. A template language is nothing but a mix of HTML and simple programming logic.**

02:23- Updating Settings For Template Creation:

First of all, it is necessary to update the settings.py file with the name of the template directory. All the templates that we will create in the future will be contained in this directory.

Updating settings.py file :

Step 1: Open settings.py

Step 2: Find 'DIRS' in settings.py. Write "templates" inside the double quotes in DIRS. This is the folder where we will keep our html template files

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

**'DIRS': ["templates"],**

'APP\_DIRS': True,

'OPTIONS': {

# ... some options here ...

},

},

]

After updating settings, we are all set to create our first Django template.

03:40 - Creating Templates :

Step 1: Right click on the textutils folder.

Step 2: Click on 'new'.

Step 3: Click on the "Directory" and name the directory as "templates" then press Enter.

04:40 - Creating index.html :

Now, we are going to create a file named "index.html" inside the template folder. This is the template file we were talking about for so long.

Steps To Create index.html?

Step 1: Right-click on the template folder.

Step 2: Click on "New"

Step 3: Click on "HTML file" and name the file as "index.html."

To return anything from this HTML file, we need to make some changes in views.py file. Open views.py file and type the import statement given below:

from django.shortcuts import render

After importing render, replace return HttpResponse with the below code:

def index(request):

return render(request, 'index.html')

render() function takes three arguments:

request: It is the argument that is required to serve the input request.

Name of the template file: The name of the template which we want to use. Here, we have passed "index.html."

Name of the dictionary: We can also pass a dictionary containing variables to the template if we want.

Restart the server and head over to http://127.0.0.1:8000/.I hope you have followed all the steps carefully and everything is working fine at your end. This tutorial ends here, and we will continue our Django journey in the next tutorial.

**12.**

**Creating Homepage of our TextUtils Website** | Python Django Tutorials In Hindi #9

After we created a template and defined some initial endpoints to work with, it's time to create a homepage for our textutils website. The home page will contain two things :

Text Area: For taking text input from the user.

Buttons: For redirecting the user to a specific endpoint.

So, let's start with the development of textarea.

0:55 - Creating Textarea :

Before getting our hands dirty by writing code for creating a text area, it is essential to understand why do we need a text area? The purpose of our textutils website is to perform some basic operations on the text entered by users. But to perform any function on the text, we need text! So, from where will we get this text? To get the text from the user, we need to build a text area into which the user can enter the text.

I think we have talked much, and it's time to start the development of our website. Follow the steps given below to create a text area.

Steps To Create Textarea:

Step 1: Open index.html.

Step 2: Type the following code :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Template is working</title>

</head>

<body>

<h1>Welcome to the text analyazer. Enter your text below</h1>

<form action='/removepunc' method='get'>

<textarea name='text' style='margin: 0px; width: 721px; height: 93px;'></textarea>

</form>

</body>

</html>

I am assuming that you already have a firm grip on HTML. If that is not the case, then you can learn HTML within 30 minutes by clicking here.

03:08 - Creating Buttons :

We will use the HTML <button> tag for creating buttons.Open index.html and type the following code :

<button type='submit'> Analyze Text</button>

Here, we have created only one button, i.e., Analyze Text. But, we will include more buttons in the future.

03:30- Transfering Text Entered By User To A Specific Endpoint :

We have successfully created a text area and a analyze button. But, are they performing any function? You can say that we can enter text in the text area, so it is performing its function correctly. But, what about the button? On clicking the button, it takes the user to nowhere! Therefore, we will now include the action attribute in the form tag that will send the entered text to a specific endpoint when the user clicks on the Analyze Text button.

Type the following code :

<form action='/removepunc' method='get'>

<textarea name='text' style='margin: 0px; width: 721px; height: 93px;'></textarea>

<button type='submit'> Analyze Text</button>

</form>

Restart the Django development server. From now, any text entered in the text area will be sent to the removepunc endpoint. For example, I entered "Code With Harry" in the text area and then clicked on the Analyze Text button. You can see in the image given below that "Code With Harry" is sent to the removepunc endpoint.

With this, we have successfully created the home page of our website. I hope each and every step is working fine on your end.

**13**

**Django Site: Coding The Backend** | Python Django Tutorials In Hindi #10

We have come so far in our Django journey and successfully made a skeleton of our textutils website. Now, it's time to give some powers to this skeleton so that it can perform some function. In this tutorial, we will build logic to remove all the punctuations from the text entered by the user.

01:30 - **Creating Checkbuttons :**

First of all, we are going to make some changes to the pipeline that we have laid in our previous tutorials. Now, we will use only one view function named analyze. So, open urls.py file and type the below code :

from django.contrib import admin

from django.urls import path

from . import views

urlpatterns = [

path('admin/', admin.site.urls),

path('', views.index, name='index'),

path('analyze', views.analyze, name='analyze')

]

In views.py file, delete all the view functions that we made earlier and define only one function, called analyze. Type the following code:

def analyze(request):

#Get the text

djtext = request.GET.get('text', 'default')

Now, open index.html file and type the following code for creating checkbox:

**<input type="checkbox" name="removepunc"> Remove Punctuations<br>**

Restart your server and check whether all the things are working correctly or not.

06:20 - Creating A New Template:

Now, we will create a new template named "analyze.html."

Steps To Create analyze.html:

Step 1 : Right click on the templates folder.

Step 2 : Click on "New" and select the HTML file.

Step 3 : Name the file as analyze.html and press enter. Now, open analyze.html and type the below code :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Analyzing Your Text...</title>

</head>

<body>

<h1>Your Analyzed Text - {{ purpose }}</h1>

<p>

{{ analyzed\_text }}

</p>

</body>

</html>

**In the above code, we have created a <h1> tag into which we have passed a variable named "purpose." We have also made a paragraph tag into which we have passed another variable called "analyzed\_text."**

07:30 - Declaring analyzed\_text and purpose variable :

Open views.py and type the following code:

def analyze(request):

# Get the text

djtext = request.GET.get('text', 'default')

analyzed=djtext

params = {'purpose': "Removing Punctuations", 'analyzed\_text': analyzed}

return render(request,'analyze.html',params)

Let's understand the above code line by line:

**First of all, we have stored the text entered by the user in the djtext variable.**

**Afte that, we have stored the same string in the analyzed variable so that we can use this value in the params dictionary.**

**Then, we have created a dictionary named params which holds the key-value pair for purpose and analyzed\_text variable that we used earlier in analyze.html.**

In the last line, we have used the render statement to return the text from the analyze.html file.

Now, let's check whether everything is working correctly or not. Restart the server and type anything in the text area. I wrote "Django Is Easy" and then clicked on the Analyze Text button. In the image given below, you can see that the same text is sent to the removepunc endpoint with the help of analyze.html.

09:25 - Logic Building For Removing Punctuations :

Now, we will write code for removing punctuations marks from the text entered by the user. Open views.py file and type the code given below:

def analyze(request):

# Get the text

djtext = request.GET.get('text', 'default')

removepunc=request.GET.get('removepunc','off')

if removepunc == "on":

punctuations = '''!()-[]{};:'"\,<>./?@#$%^&\*\_~'''

analyzed = ""

for char in djtext:

if char not in punctuations:

analyzed = analyzed + char

params = {'purpose': 'Removed Punctuations', 'analyzed\_text': analyzed}

return render(request, 'analyze.html', params)

else:

return HttpResponse('Error')

Let's understand the above code one line at a time. First of all, we are checking if the removepunc checkbox is on or off. If it is off, then the control will come to the else statement, and an error message will be printed on the user's screen. Otherwise, the if block of the code will be executed. Inside the if block,we have stored all the punctuation marks in a punctuations variable and made an empty variable named analyzed. When the user checks on the removepunc checkbox, iteration will be performed on the user's text with the help of the for loop. Finally, all the punctuations will be removed from the text entered by the user.

Example: I wrote the text "Remove;';' all ;./ the punctuations;#" and then clicked on Analyzed Text button after checking on the removepunc checkbox. In the image given below, you can see that all the punctuations are removed.

YOU'VE DONE IT. YOU'VE MADE YOUR VERY FIRST WORKING WEBSITE!!! I'm so proud of you. We will continue our Django series in the next tutorial. I hope you all are falling in love with Django ❤

**14**

**Adding More Features To TextUtils Website | Python Django Tutoria**ls In Hindi #12

In our previous tutorial, we have made a removepunc endpoint that removes all the punctuations from the text entered by the user. Now, it's time to include some other exciting functionalities on our website.

01:30 - Logic Building For UPPERCASE:

We will include a functionality named UPPERCASE that will convert the user's text into the uppercase. First of all, we need a checkbox like we created for removepunc. Open index.html and type the following code:

<input type="checkbox" name="fullcaps"> UPPERCASE<br>

Now, open the views.py file and type the following code:

fullcaps=request.GET.get('fullcaps','off')

In the above code, we have made a variable named fullcaps which stores the on or off value of the fullcaps checkbox. Now, type the following code in views.py :

elif fullcaps=="on":

analyzed=""

for char in djtext:

analyzed=analyzed+char.upper()

params = {'purpose': 'Change To Uppercase', 'analyzed\_text': analyzed}

return render(request, 'analyze.html', params)

else:

return HttpResponse('Error')

Let's understand the above code line by line. We have used the elif loop to check whether the fullcaps checkbox is on or off. After that, we have made an empty string variable named analyzed. When the user checks on the fullcaps checkbox, then the for loop inside the elif loop will be executed, and an iteration will be performed on djtext. On performing iteration, the text will be converted into the upper case with the help of the upper() function. Example: I wrote "change to upper case" and checked on the UPPERCASE check button. You can see in the image below that the text is converted into upper case.

06:10 - Logic Building For newlineremover :

Now, we will write code for removing a new line from the text entered by the user. Open index.html and type the following code:

<input type="checkbox" name="newlineremover"> New Line Remover<br>

Open views.py and type the following code :

elif newlineremover=="on":

analyzed=""

for char in djtext:

if char!="\n":

analyzed=analyzed+char

params = {'purpose': 'Removed NewLines', 'analyzed\_text': analyzed}

# Analyze the text

return render(request, 'analyze.html', params)

In the above code, we are using the same approach that we have used for creating other buttons. We are using an elif loop to check whether the newlineremover checkbox is turned on or off by the user. When the user checks on the newlineremover, then the for loop will be executed, and the new line will be removed with the help of a newline character("\n").

09:20: Logic Building For extraspaceremover :

Isn't it irritating when we add extra spaces by mistake while writing an important mail or assignment? How would it be if you can remove all the extra spaces with your own website? So let's write code for removing additional spaces from the text entered by the user. Open index.html and type the following code for creating checkbox:

<input type="checkbox" name="extraspaceremover"> Extra Spaces Remover<br>

Open views.py and write the following code :

elif(extraspaceremover=="on"):

analyzed = ""

for index, char in enumerate(djtext):

if not(djtext[index] == " " and djtext[index+1]==" "):

analyzed = analyzed + char

params = {'purpose': 'Removed NewLines', 'analyzed\_text': analyzed}

# Analyze the text

return render(request, 'analyze.html', params)

In the above code, we have used elif to check whether the extraspaceremover is on or off. If the checkbox is on, then the for loop will be executed, and extra spaces will be removed with the help of the built-in function enumerate(). For example, I wrote, "Remove all the extra spaces." I checked on the Extra Spaces Remover button. You can see in the image below that all the extra spaces are removed.

With this, we have successfully added some amazing functionalities to our textutils website. This tutorial ends here, and I will see you in the next tutorial.

**15.**

**Adding Boostrap To Our Django Website | Python Django Tutorials In Hindi #14**

In our previous tutorials, we have added some really amazing functionalities to our website. But our website looks very simple! Till now, we have just focused on adding different features on our website. But, now it's time to focus on the design of our website. It's time to make our site beautiful. But how are we going to do that? We are going to use Bootstrap to change the look of our website. And trust me! Adding Bootstrap is the best thing that you can do to kickstart the designing of a website. So, let's start with the introduction of Bootstrap.

What Is Bootstrap?

Bootstrap is a front-end development framework.

It is a library of HTML, CSS, and JS, which is used to create modern websites and web applications.

Sounds great! But, you might be thinking that you don't know anything about it, so how you will use it. Don't worry! Click here to learn Bootstrap in one hour.

02:05- Adding Bootstrap Starter Template To Our Website :

Let me send you to the Getting Started page of Bootstrap. Scroll down and copy the starter template

After copying the starter template, open index.html file, and paste the starter template. You can also copy the starter template from below :

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">

<title>Hello, world!</title>

</head>

<body>

<h1>Hello, world!</h1>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI" crossorigin="anonymous"></script>

</body>

</html>

02:40- Adding Navigation Bar :

After adding a starter template, it's time to include the navigation bar on our website. Click here and copy the navbar of your choice. Below is the code for the one that I am using :

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="#">Navbar</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Link</a>

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">

Dropdown

</a>

<div class="dropdown-menu" aria-labelledby="navbarDropdown">

<a class="dropdown-item" href="#">Action</a>

<a class="dropdown-item" href="#">Another action</a>

<div class="dropdown-divider"></div>

<a class="dropdown-item" href="#">Something else here</a>

</div>

</li>

<li class="nav-item">

<a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>

</li>

</ul>

<form class="form-inline my-2 my-lg-0">

<input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

</form>

</div>

</nav>

After copying the code, paste this code in index.html. After pasting your website will look something like this :

**04:40 - Adding Alerts :**

Click here to select the alert of your choice. I am using dismissible alerts and below is the code for same :

<div class="alert alert-success alert-dismissible fade show" role="alert">

<strong>Welcome to TextUtils!</strong> You can do anything with your text here!

<button type="button" class="close" data-dismiss="alert" aria-label="Close">

<span aria-hidden="true">×</span>

</button>

</div>

06:00- Adding Text Area :

Copy and paste the code given below to add text area:

<div class="form-group">

<label for="exampleFormControlTextarea1">Enter your text here and let Text Utils do the magic</label>

<textarea class="form-control" name='text' id="exampleFormControlTextarea1" rows="9"></textarea>

</div>

09:00 - Adding Switches :

Copy the below code and paste in the index.html file.

<div class="custom-control custom-switch">

<input type="checkbox" name="removepunc" class="custom-control-input" id="customSwitch1">

<label class="custom-control-label" for="customSwitch1">Remove Punctuations</label>

</div>

Use the same code to make more switches. But, do not forget to change the switch id.

12:20 - Adding Analyze Text Button :

Type the code given below to add the analyze text button :

<button type="submit" class="btn btn-dark mt-2">Analyze Text</button>

After successfully adding buttons and switches, add action attribute in the form tag. Write the following code :

<form action='/analyze' method='get'>

With this, we have successfully added Bootstrap to the home page of our textutils website. But, we have not included Bootstrap on the endpoints pages of our website. So, let's add Boostrap on the endpoint pages.

15:00 - Adding Bootstrap To analyze.html:

Copy the below code and paste it in analyze.html file :

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css" integrity="sha384-GJzZqFGwb1QTTN6wy59ffF1BuGJpLSa9DkKMp0DgiMDm4iYMj70gZWKYbI706tWS" crossorigin="anonymous">

<title>Text Utils</title>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<a class="navbar-brand" href="#">TextUtils.in</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNavDropdown">

<ul class="navbar-nav">

<li class="nav-item active">

<a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">About Us</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Contact Us</a>

</li>

</ul>

</div>

<form class="form-inline">

<input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

</form>

</nav>

<div class="alert alert-success alert-dismissible fade show" role="alert">

<strong>Success!Your Text Has Been Analyzed</strong>

<button type="button" class="close" data-dismiss="alert" aria-label="Close">

<span aria-hidden="true">×</span>

</button>

</div>

<div class="container mt-5">

<h1>Your Analyzed Text -- {{purpose}}</h1>

<p>

{{ analyzed\_text}}

</p>

</div>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min.js" integrity="sha384-wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/yoi7FRNXMRBu5WHdZYu1hA6ZOblgut" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js" integrity="sha384-B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k" crossorigin="anonymous"></script>

</body>

</html>

In the above code, we have added the starter template and navigation bar of the bootstrap. After that, we have passed the analyzed\_text and purpose variable inside the div container.

And that's how we completely changed the overall look of our website without even writing a single line of code. Isn't it amazing? That's why I love bootstrap, and you should too! This tutorial ends here, and we have also completed the development of our textutils website.

**16.**

**Fixing Bugs In Our TextUtils Website | Python Django Tutorials In Hindi #15**

In our previous tutorial, we successfully created our textutils website and added bootstrap to it. But, while using our website, I found out that there is a bug on our website. In this tutorial, we will fix all the bugs on our textutils website.

Let's have a look at the bug I was talking about. I entered some text in the text area and turned on the UPPERCASE button. Here is the text that I entered :

In the image given below, you can see that the text that I entered was changed to uppercase. But, all the new lines were also removed automatically. So, why are we facing this issue? I would suggest you spend some time in finding out the reason of this bug. We will discuss the cause and solution of this bug further in this tutorial.

If you have figured out the reason for this bug, then I must say that you are a good programmer, and all the basic concepts are crystal clear to you. Now, let's discuss the cause:

This error is coming from our analyze.html file. In our analyze.html file, we have directly dumped the analyzed\_text variable inside the paragraph tag. But, it is the default property of HTML to chop off the new line character inside the paragraph tag. To stop this, HTML requires us to use a <pre> tag, which indicates that the text is preformatted.

**Open analyze.html and add the <pre> tag :**

<div class="container mt-5">

<h1>Your Analyzed Text -- {{purpose}}</h1>

<p>

<pre> {{ analyzed\_text}} </pre>

</p>

</div>

Now, let's check whether everything is working correctly or not. Again, I entered the text and turned on the UPPERCASE button. Now, you can see in the image given below that newlines are still there, and HTML is not removing them automatically.

And that's how we quickly fixed the bug on our textutils website. I hope all the concepts are crystal clear to you. Ask your queries in the QnA section.

**17.**

**Django CSRF Tokens & Post Request | Python Django Tutorials In Hindi #16**

In this tutorial, we will discuss the concepts of HTTP GET and POST method. We will also discuss the Django CSRF tokens. So, before starting our discussion on GET and POST request, take a look at the image given below :

In the above image, I have typed some text and changed it to uppercase.

Output:

As expected, the text is changed to uppercase. Now, notice the highlighted part of the URL. You can see that the text entered is exposed in the URL. So the following questions arise here :

Why the input text exposed in the URL?

What if we do not want to show the text in the URL?

Now, I will answer the above questions one by one. So let's start with question 1.

Why the input text exposed in the URL?

Ans: The text is visible in the URL because we are using the GET method for passing the information. HTTP is a request-response protocol that takes the request from the user, and then a server returns the response to the user. So, to establish this request-response connection, we need some methods. The two most commonly used HTTP methods are HTTP GET and POST. So, let's understand the GET method :

HTTP GET :

The GET method is the default submission method for a form.

The GET method sends the data in the form of URL parameters. Therefore, any data sent with the help of the GET method remains visible in the URL.

Since the data is exposed in the URL, the GET method is not considered to send sensitive information such as passwords.

The GET method reveals the data in the URL bar; therefore, the length of the URL increases. The maximum length of a URL is 2048 characters, so only a limited amount of data can be sent using the GET method. The following error occurs when we try to send more than 2048 characters using GET :

Now, we are done with the GET method details, and it's time to answer Que 2.

What if we do not want to show the text in the URL?

Ans: In case we do not want to expose the data in the URL bar, then we can use the POST method instead of GET. Let's start our discussion on the POST method :

POST :

Data sent by the POST method never gets visible in the URL box, and therefore it is more secure than the GET method, and sensitive information can be sent with the help of this method.

Since the data is not visible in the URL query, the length of the URL remains less than 2048 characters, and a large amount of data can be sent with the help of the POST method.

Data is sent to the server in the form of packages in a separate communication with the processing script.

Now, we will start our discussion on CSRF tokens.

WHAT ARE CSRF TOKENS?

**CSRF stands for Cross-Site Request Forgery.**

**The server-side application generates and transmits a huge, random, and unpredictable number to the client to make sure that the request is coming from the original client and not from a malicious website.**

**CSRF tokens are used to protect the site against CSRF attacks.**

How to include CSRF token to your website :

Type the following code in index.html :

<form action='/analyze' > {% csrf\_token%}

With this, you have successfully included CSRF token to your website. Now, we will see how to use the POST method on our website.

Type the following code in index.html :

<form action='/analyze' method='post'> {% csrf\_token %}

After this, replace all the request.GET.get with the request.POST.get. With this, we have successfully changed the HTTP method of the textutils website. Look at the below image :

You can see in the above image that the user's data is no longer exposed in the URL box. This tutorial ends here, and I will see you in the next tutorial**.**

**18**

**Wrapping Up TextUtils Website |** Python Django Tutorials In Hindi #18

We have completed the development of the textutils website, and in this tutorial, we will push this project to Github. Before pushing, make sure that you remove all the redundant things from the project such as duplicate files or piece of code, unnecessary print statements etc. After removing all the useless head over to Github and create a new repository and then follow the below steps to push textutils on Github. If you have no prior experience of Git and Github, then click here to learn Git in one video. You can also access this playlist to learn Git from basics to advance.

Step 1: Open Windows Powershell.

Step 2: Get into the directory containing the textutils project. You can also use the below code in case your textutils project is also inside the PycharmProjects folder.

cd ./PycharmProjects/textutils

Step 3: Now cd into the textutils folder :

cd .\textutils\

Step 4: Initialize the local directory by typing the below command :

git init

Step 5: Add files to your local directory :

git add .

Step 6: Commit the files by typing the below command :

git commit -m "initial commit"

Step 7: Copy the URL of your repository.

Step 8: Type the below command :

git remote add origin remote repository URL

git remote -v

Step 9: Push the changes to Github.

git push -f origin master

With this, we have successfully pushed our textutils website to Github.

Code as described/written in the video

Go to https://github.com/haris989/TextUtils for information on TextUtils website and its further development. Make sure to contr