How to use URL Validator in Django?

Last Updated : 29 Sep, 2023

In Django, a popular Python web framework, URL validation can be easily implemented using built-in tools and libraries. In this article, we will explore how to use the URL validator in Django to ensure that the URLs in your web application are valid and secure.

Django's URL Validator

Django provides a URL validator as part of its built-in validation tools. This validator checks whether a given string is valid according to the URL specification. To use this validator, follow these steps:

Step 1: Create a project folder, and Navigate to the Project Directory

```
django-admin startproject bookstore
cd bookstore
```

Activate Virtual Environment (Optional)

Step 2: Create a Django App

Inside your project, create a Django app named "mini":

```
python manage.py startapp mini
```

Step 3: Define the Model

In this example, we'll create a simple model to store the validated URLs. Open the models.py file in your url_validator_app folder and define the model as follows:

Python3

```
# mini/models.py
from django.db import models

class ValidatedURL(models.Model):
    url = models.URLField(unique=True)

    def __str__(self):
        return self.url
```

Step 4: Create form.py

By defining this form class, we've created a structure for capturing and validating URL input from users. It encapsulates the logic needed for URL validation, making it easy to use in our views and templates.

Python3

```
# mini/forms.py
from django import forms
from django.core.validators import URLValidator

class URLForm(forms.Form):
    url = forms.URLField(
        label='Enter a URL',
        validators=[URLValidator()],
        widget=forms.TextInput(attrs={'placeholder': 'https://example.com'})
    )
```

Step 5: Genrate view of the App

The index view handles URL validation and form submissions, while the success view displays the list of validated URLs.

Python3

```
# mini/views.py
from django.shortcuts import render, redirect
from .forms import URLForm
from .models import ValidatedURL

def index(request):
    if request.method == 'POST':
        form = URLForm(request.POST)
        if form.is_valid():
            url = form.cleaned_data['url']
            ValidatedURL.objects.create(url=url)
            return redirect('success')
    else:
        form = URLForm()
    return render(request, 'url_validator_app/index.html', {'form': form})

def success(request):
    validated_urls = ValidatedURL.objects.all()
    return render(request, 'url_validator_app/success.html', {'validated_urls': va
```

Step 6: Create the Templates

Create two HTML templates: one for the form and another for displaying the validated URLs. Create a templates folder within your app directory and add the following templates:

template/index.html: URL Validator

HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>URL Validator</title>
</head>
<body>
    <h1>URL Validator</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Submit</button>
    </form>
```

```
</body>
</html>
```

template/index2.html: Validated URLs

HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Validated URLs</title>
</head>
<body>
    <h1>Validated URLs</h1>
    <ul>
        {% for url in validated_urls %}
            <li>{{ url }}</li>
        {% empty %}
            <li>No validated URLs yet.</li>
        {% endfor %}
    </ul>
    <a href="{% url 'index' %}">Back to validation</a>
</body>
</html>
```

Step 7: Configure URLs in the mini/urls.py

Configure your app's URLs by creating a urls.py file within the app folder:

Python3

```
# mini/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('success/', views.success, name='success'),
]
```

Step 8: Incude app URLs in the Project URLs

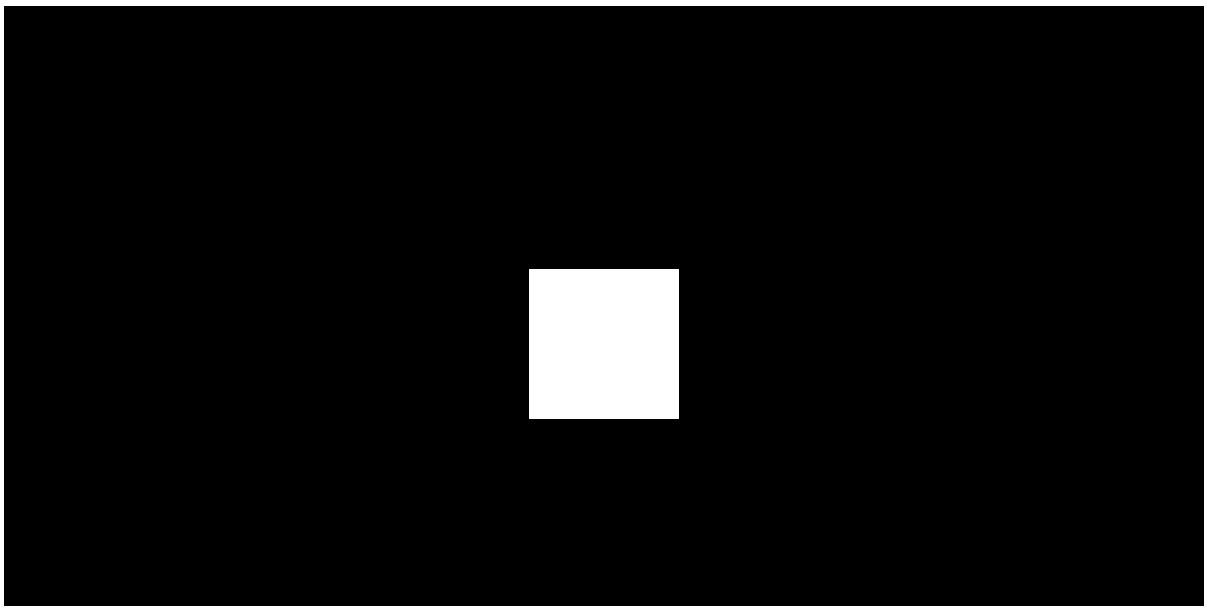
urls.py: In your project's urls.py, include the URLs from your app:

Python3

```
# url_validator_project/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('mini.urls')),
]
```

Output:



Are you ready to elevate your web development skills from foundational knowledge to advanced expertise? Explore our Mastering Django Framework - Beginner to Advanced Course on GeeksforGeeks, designed for aspiring developers and experienced programmers. This comprehensive course covers everything you need to know about Django, from the basics to advanced features. Gain practical experience through **hands-on projects** and real-world

applications, mastering essential Django principles and techniques. Whether you're just starting or looking to refine your skills, this course will empower you to build sophisticated web applications efficiently. Ready to enhance your web development journey? Enroll now and unlock your potential with Django!

P puja2...

Next Article

How to use Regex Validator in Django?

Similar Reads

How to use Regex Validator in Django?

Django, a popular web framework, provides a built-in regex validator that allows you to enforce specific patterns on input data. Whether you need to validate use...

3 min read

How to Add URL Parameters to Django Template URL Tag?

When building a Django application, it's common to construct URLs dynamically, especially when dealing with views that require specific parameters. The Djang...

4 min read

Comparing path() and url() (Deprecated) in Django for URL Routing

When building web applications with Django, URL routing is a fundamental concept that allows us to direct incoming HTTP requests to the appropriate vie...

8 min read

Django URL patterns | Python

Prerequisites: Views in Django In Django, views are Python functions which take a URL request as parameter and return an HTTP response or throw an exceptio...

2 min read

url - Django Template Tag

A Django template is a text document or a Python string marked-up using the Django template language. Django being a powerful Batteries included...

3 min read

URL fields in serializers - Django REST Framework

In Django REST Framework the very concept of Serializing is to convert DB data to a datatype that can be used by javascript. Every serializer comes with some...

5 min read

Build a URL Size Reduce App with Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. In this article, we will learn to build a...

5 min read

Create URL Bookmark Manager Using Django

This article explains how to make a tool called a Bookmark Organizer with Django for a website. With this tool, we can add, create, update, delete, and edit...

5 min read

Get the Current URL within a Django Template

Django, a high-level Python web framework, encourages rapid development and clean, pragmatic design. One common requirement when developing web...

3 min read

Get the Absolute URL with Domain in Django

When developing web applications, generating the full URL (including the domain) is often necessary. For instance, sending confirmation emails with links,...

3 min read

Article Tags : Geeks Premier League | Project | Python | Geeks Premier League 2023 | +1 More

Practice Tags : python

Corporate & Communications Address:- A-143, 9th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) | Registered Address:- K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305