Django URL Dispatcher Tutorial

Last Updated : 24 Sep, 2024

In the Django application URL Dispatcher is used to map the URL with view, which in turn helps to handle the request and produce a response. In this article, we will learn all the concepts related to the Django URL Dispatcher.

Methods of Using Django URL Dispatcher

- Creating Url Patterns in Django
- Using Regular Expression Captures in Django URL Patterns
- Naming URL Patterns in Django
- Inverting URL Patterns in Django
- Using Namespaces in Django URL Patterns
- Using Class-Based Views in Django URL Dispatcher

Creating Url Patterns in Django

In Django, URL patterns are the fundamental concept that is used to map URLs (Uniform Resource Locators) to views, by which we can specify how different URLs in your web application should be handled. URL patterns are defined in your Django project's urls.py file. Let us understand with this example:

To create a URL Pattern we should

1. open the urls.py file in our Django app
2. Import the path() function from django.urls.
3. Define a list of URL patterns

In this example, we've defined two URL patterns, '/home/' and '/about/', and specified which view functions should handle these URLs.

Python

```
from django.urls import path
from . import views

urlpatterns = [
    path('home/', views.home_view),
    path('about/', views.about_view),
]
```

Understanding **Django's URL dispatcher** is fundamental for **routing in web applications.** To explore this and other advanced Django topics, the **Django Web Development Course** will help you achieve mastery.

Using Regular Expression Captures in Django URL Patterns

If we want to extract the value from urls then pass it to views than we can do it with the help of regular expression captures .The extracted value then can be used as according to the need in the view function.

To use the Regular Expression capture follow the following steps:

1. Define the regular expression in url pattern and add capturing groups by the help of '**()**' to collect values from the pattern.
2. In the view function, include for each capturing group.
3. When the url pattern is matched ,the matched view function is called.

In below code the 3rd URL pattern is dynamic and captures an integer value called blog_id from the URL. For example, if you access http://yourdomain.com/blog/1/, it will display the detail page for the blog post with an ID of 1. The name of this URL pattern is 'blog_detail'.

Python

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('blog/<int:blog_id>/', views.blog_detail,
name='blog_detail'),
```

```
7
8 ]
```

views.py: The view.py code for the above urls is shown below. In the below code **'home'** is a basic view function. The **'blog_detail'** is a view function in which the value captured value from url is passed as argument here. The argument here is **'blog_id' .** This can be used inside our view function according to the need.

Python

```python
from django.shortcuts import render
from django.http import HttpResponse


def home(request):
    return HttpResponse("Welcome to our website!")

def blog_detail(request, blog_id):
    blog_post = {'id': blog_id, 'title': 'Sample Blog Post', 'content': 'This is the content of the blog post.'}
    context = {'blog_post': blog_post}
    return render(request, 'blog_detail.html', context)
```

Naming URL Patterns in Django

We can name our url pattern in Django so that it makes our work easy while refering to them in our code and html files.

To give a name to the url pattern we should follow the following steps:

1. In the path function specify the name ,with a string value.
2. In the templates, use the {% url %} and in the code use the reverse() function to refer to the named URL pattern.

Let us understand with this example:

In this Django URL configuration, we have assigned names to two URL patterns: **'home'**, **'about'.** These names help us refer to these URLs in our code,

making it more readable and maintainable.

Python

```python
from django.urls import path
from . import views

urlpatterns = [
    path('home/', views.home, name='home'),
    path('about/', views.about, name='about'),
]
```

Use in HTML template

We can use the name of the url pattern as :

```
{% url 'home' %}
```

For Example:

HTML

```html
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h2>Welcome To GFG</h2>
<a href="{% url 'home' %}">HOME</a>
</body>
</html>
```

Inverting URL Patterns in Django

In Django, inverting URL patterns refers to the process of reversing a URL to its corresponding view or URL pattern name. This is particularly useful when we

want to generate URLs dynamically in our views or templates without hardcoding them.

We can invert URL patterns in Django using two primary methods:

- **Using the {% url %} template Tag**

In Django templates, we can use the **{% url 'url_name' %}** template tag to reverse a URL based on its name. Here, **'url_name'** should match the name assigned to the URL pattern in your urlpatterns.

```
<a href="{% url 'home' %}">Home</a>
```

- **Using the reverse() function in views**

In your Python code (views, models, etc.), you can use the reverse() function to reverse a URL based on its name. Import it from django.urls and pass the URL pattern name as an argument. This will generate the URL for the 'home' URL pattern and perform a redirect.

Views.py

Python

```python
from django.urls import reverse
from django.shortcuts import redirect

def my_view(request):
    return redirect(reverse('home'))
```

Using Namespaces in Django URL Patterns

In Django, namespaces are used to organize URL patterns and prevent naming conflicts when multiple applications are integrated into a single project. Namespaces allow us to group URLs under a specific namespace or prefix, making it easier to manage and distinguish different parts of your application.

Example: Here is how we can use namespaces in Django URL patterns:

In the project's main urls.py file, include your app's URLs using the include() function with the namespace parameter.

Python

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('myapp1/', include('myapp1.urls', namespace='myapp1')),
    path('myapp2/', include('myapp2.urls', namespace='myapp2')),
]
```

To reference URLs with namespaces in your templates or views, use the {% url %} template tag or the reverse() function with the format '**namespace:url_name**'. For example:

```
<a href="{% url 'namespace:url_name'' %}">Home</a>
```

Class-Based Views in Django URL Dispatcher

In Django, you can use class-based views (CBVs) to handle URL routing and view logic. Class-based views provide a more organized and reusable way to structure your views compared to function-based views .

Creating a Class-Based View:

To create a class-based view, define a Python class that inherits from one of Django's provided generic view classes or create your own custom class. For example, let's create a simple class-based view for displaying a list of items:

Example: In this example, we've created a class called ItemListView that inherits from View and defines a get method to handle GET requests.

Python

```python
from django.views import View
from django.http import HttpResponse

class ItemListView(View):
    def get(self, request):
        items = ["Item 1", "Item 2", "Item 3"]
        return HttpResponse("\n".join(items))
```

Mapping the Class-Based View to a URL

To map your class-based view to a URL, you can use the as_view() method of your view class when defining URL patterns in your app's urls.py file:

Here, we've associated the ItemListView class-based view with the URL pattern 'items/' and given it the name 'item-list'.

Python

```python
from django.urls import path
from .views import ItemListView

urlpatterns = [
    path('items/', ItemListView.as_view(), name='item-list'),
]
```

So, Now you have the knowledge about the URL dispacting in Django

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!

Previous Article

How to build a URL Shortener with Django ?

Next Article

Similar Reads

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

How to use URL Validator in Django?

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 ho...

3 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 : Django Python Django-basics Django-Projects +2 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