How to Add Search to a Django App

Integrating search into a Django application is not difficult. Here are step-by-step instructions.

Integrating **search into a Django application** is not difficult. Here are step-by-step instructions.

Adding a search feature to a web application allows users to navigate it easily by searching for what they want. Django provides built-in support for search functionality using its powerful ORM and query engines. With Django, you can implement different types of searches, including keyword search, simple search, and advanced search with filters.

Implement search functionality in Django

Django allows you to perform different types of searches using built-in methods and functions. You can search by simple keywords or use advanced search. Advanced search is recommended if you have a complex application, such as an e-commerce web, while simple keyword searches are often preferable for complex projects.

Implement simple search in Django

To create a simple search feature, you should start by building a search bar. You can do this by creating it in the navbar. Bootstrap provides a built-in navbar template with a search bar. You can easily integrate Bootstrap and its components into your Django project. Create a search bar in the HTML file, set the form method to POST and set the name attribute for the input field as follows:

```
?{% csrf_token %} ? ?
```
Search

In the above code, the name of the input field is search_query. This form uses Django's CSRF token to prevent CSRF attacks. To get the search bar working, follow these steps:

Create a viewer for search

1. Open the **views.py** file and import the template from the **models.py** file :

```
from .models import ModelName
```

1. Create a view function for the search feature:

```
def search_feature(request): ????# Ki?m tra xem truy v?n có ph?
i request post ????if request.method == 'POST': ????????# Truy xu?t truy v?
n tìm ki?m c?a ng??i dùng ????????
search_query = request.POST['search_query'] ????????#L?c model theo truy v?
n tìm ki?m ????????
posts = Model.objects.filter(fieldName__contains=search_query) ????????
return render(request, 'app/template_name.html', {'query':search_query, 'posts':p
????else: ????????return render(request, 'app/template_name.html',{})
```

The above function first checks if the client is sending a POST query. If the test is successful, it will continue to retrieve the value of the search query from the user as follows:

```
search_query = request.POST['search_query']
```

After retrieving the value of the search query from the user, this function filters the sample containing it using the **_contains** method (case-insensitive). You should follow this format:

```
fieldName__contains
```

For example, if you want users to search based on the name form field **,** you should edit the code like this:

```
name__contains=search_query
```

Finally this function outputs a template and passes the search query and the filtered template as context.

However, if the method of this form is not a POST query , the function outputs a form with an empty dictionary and does not process the search query.

Create a template for search results

1. Create an HTML file to return search results to the client.

1. Export search results on the page for users to see. The code in the HTML file will look like this:

```
{% if query %} ???? ???????? ???????? ????????{% for post in posts %}
??????????? ???????????????? ????????????????
```

{{post.title}}

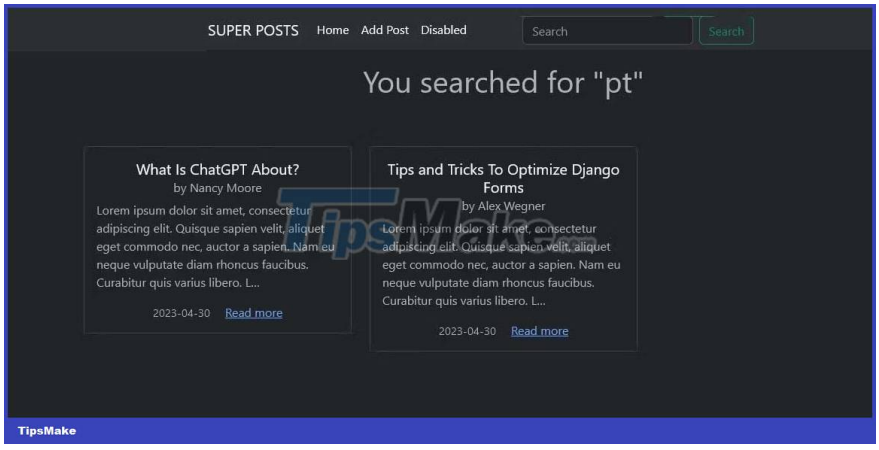
```
 ??????????? ????????{% endfor %} ???????? ???? {% else %} ???? ????
```

Please enter a search query

```
 {% endif %}
```

The above HTML sample checks to see if the user entered a search query in the search bar. If the user enters a search query, **the for loop** iterates over the results and returns them to the user. When there is no search query, a message is displayed for the user to enter the query.

The absence of a search query is when your users go directly to the URL without entering content in the search bar. You need to make sure you're using Django's template inheritance in the HTML file.



1. Edit the HTML code to return an error message if there are no search results.

```
{% if query %} ???? ???????? ???????? ????????{% if posts %} ???????????
???????????{% for post in posts %} ???????????????? ??????????????????
??????????????????
```

{{post.title}}

```
 ???????????????? ???????????{% endfor %} ??????????? ????????{% else %}
???????????
```

No search results found

```
 ????????{% endif %} ???????? ???? {% else %} ????
```

Please enter a search query

```
 {% endif %}
```

New HTML template for better user experience. It introduces a conditional statement to check if a connection is already available in the database. If yes, it shows the result. Otherwise, it sends an error message to the user.

URL pattern configuration

1. If this doesn't already work, create a file urls.py in the application directory.
2. At **urls.py** , the file creates a URL pattern for the search page:

```
from django.urls import path from . import views urlpatterns = [ ????
path('search/', views.search_feature, name='search-view'), ]
```

The above program first enters the path function and the file views associated with the application. Then it generates the search-view name path for the search page.

Add form creation to the search bar. Its URL will point to the dedicated search URL. In this case, the form points to **search-view** .

```
 ??? ???
```
Search

If there is no form action pointing to the search URL, your search will not work. Remember that the search URL must point to the Django viewport that handles the logic of the search feature.

Here's **how to add search to your Django app** . Hope the article is useful to you.