django-table-sort

Eduardo Leyva

Sep 17, 2023

USER GUIDE

Create tables with sorting links on the headers in Django templates.

CHAPTER

ONE

INSTALLATION

1.1 1. Install the package

You can install *django-table-sort* via pip from PyPI:

```
$ pip install django-table-sort
```

1.1.1 Requirements

- Python 3.7+
- Django 3.0+

1.2 2. Add *django-table-sort* to your `INSTALLED_APPS`

```
INSTALLED_APPS = [
    # ...,
    "django_table_sort",
    # ...,
]
```

1.3 3. Add the CSS file to your templates

```
<! -- Font Awesome 6 for the table icons -->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/
↪css/all.min.css" integrity="sha512-
↪1sCRPdkRXhBV2PBLUdRb4tMg1w2YPf37qatUFeS7zlBy7jJI8Lf4VHwWfZZfpXtYSLy85pkm9GaYVYMfw5BC1A==
↪" crossorigin="anonymous" referrerpolicy="no-referrer" />
<! -- css to use the header sort icons -->
<link rel="stylesheet" href="{% static 'django_table_sort.css' %}"/>
```

CHAPTER

TWO

GETTING STARTED

2.1 The Test Models

For example purposes, we'll use a simplified book app. Here are our models.

```
# app/models.py


class Person(models.Model):
    name = models.CharField(max_length=100, verbose_name="First Name")
    age = models.IntegerField(verbose_name="Age in Years")

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

2.2 Basic Usage

```
from django.shortcuts import render
from django_table_sort.table import TableSort
from app.models import Person


def view(request):
    table = TableSort(request, Person.objects.all())
    return render(request, "template.html", context={"table": table})
```

This is the basic usage of the table sort. You can use this to display a Queryset and also a list of items.

Note: The default text for the header when displaying data from a Queryset is the verbose_name of the field. For a list of any other object, you must set the header text using the column_names parameter.

2.3 Table CSS

You can provide the CSS classes that the table should have as shown below.

```
from django.views.generic import ListView
from django_table_sort.table import TableSort


class ListViewExample(ListView):
    model = Person
    template_name: str = "base.html"
    ordering_key = "o"

    def get_ordering(self) -> tuple:
        return self.request.GET.getlist(self.ordering_key, None)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context["table"] = TableSort(
            self.request,
            self.object_list,
            table_css_classes="table table-light table-striped table-sm",
            sort_key_name=self.ordering_key,
        )
        return context
```

2.4 Fields and Exclusion

The default behavior is to show all fields of the model. If you want to show only certain fields, you can set this in the fields parameter as follows.

```
TableSort(request, object_list, fields=["name"])
```

This code will display only the field “name” in the table. You can also set which fields you don’t want to display.

```
TableSort(request, object_list, exclude=["age"])
```

Any field you pass in the exclude parameter will not be displayed, and the others that aren’t will be.

> **Warning:** The current implementation looks for the exclude field first. So if you provide both fields and exclude, all the fields, no matter if they are in the list of fields you declared in the fields parameter, **will not be displayed**.

2.5 Customizing Fields Headers

```
TableSort(request, object_list, fields=["age"], column_names={"age": "Age"})
```

You can set a custom header for any field. For this, you can use the column_names parameter.

Warning: If you set the fields and exclude parameters to None and you provide the column_names parameter, all the fields that are given will be displayed.

2.6 Adding Extra Columns

Sometimes you may want to add a custom column to the table. You can do this using the added_columns parameter.

```
def sum(instance):
    return f"Sum {instance.age + 1}"


TableSort(
    request,
    object_list,
    fields=["age"],
    column_names={"age": "Age"},
    added_columns=[(("added_column_1", "Sum"), sum)],
)
```

The added_columns parameter takes a list of tuples following this pattern: ((field_identifier, field_header), callable_function). The field_identifier is a string value to identify the field, the field_header is used to set the text of the header, and the callable_function should be a function that takes one parameter and returns a string value. The callable_function will be called for each row, and the object that should be displayed is passed as a parameter to the function.

2.7 List of Items

For a list of items, you need to set the column_names. All the fields in the dictionary will be displayed.

```
TableSort(
    request,
    [person_1, person_2],
    fields=None,
    column_names={"age": "Age"},
)
```

Note: You can use the added_columns parameter to add other custom columns in the same way.

2.8 Primary Key

Sometimes you may want to show the primary key of your model. The default behavior is not to display the primary key of a Queryset since it is often not useful to show this to the user.

```
TableSort(
    request,
    object_list,
    show_primary_key=True,
)
```

2.9 Fields Order

You can set the order to display the fields in the table. For this, you should usethe field_order parameter.

```
TableSort(
    request,
    object_list,
    field_order=["age"],
)
```

This will display the "age" as the first column in the table.

Note: The fields will be displayed following the order you give, but if you don't include a given field, it will be displayed as the last. The field_order parameter works as a priority list.

2.10 Customizing the Table Template

You can customize the template used for generating the table by providing a different *template_name* parameter in the TableSort constructor. By default, the template used is *'django_table_sort/table.html'*. Here's an example:

```
TableSort(
    request,
    object_list,
    template_name="custom_template.html",
)
```

In the above example, the *'custom_template.html'* file will be used instead of the default template for generating the table.

To create your custom template, you can copy the contents of the default template *'django_table_sort/table.html'* and modify it according to your needs.

To see the different options you can provide, please see the section *TableSort*.

CHAPTER

THREE

TABLES

3.1 TableSort

class django_table_sort.table.**TableSort**(*request: HttpRequest*, *object_list: QuerySet | list*, *fields: list = ['__all__']*, *exclude: list = None*, *column_names: None | dict[str, str] = None*, *field_order: None | list[str] = None*, *sort_key_name: str = 'o'*, *table_css_clases: str = 'table'*, *table_id: str = None*, *template_name: str = 'django_table_sort/table.html'*, ***kwargs*)

Class to generate the table with the sort.

Parameters

- **request** – current `HttpRequest` to get the url lookups to create the links.
- **object_list** – `QuerySet` or `list` to fill the table.
- **fields** – `list` This field sets which fields should be displayed, the default value is ["__all__"] that will display all the fields in the model and the verbose_name of them as the header of the columns. You can use the column_names param to customize the headers.
- **exclude** – `list` Similar to the fields param, defines which fields should be excluded, all the field that aren't in the exclude list will be displayed.
- **column_names** – `dict` containing the pair {field_name: field_header}, this field has two uses, if you provide a `list` of X items this field will set which field will be displayed and the proper headers, if you provide a `Queryset` instead this field will define how the columns header will be displayed.
- **field_order** – `list` containing the fields in the order that you want
- **sort_key_name** – `str` for the key name that will be used to create the sort lookup in the urls.
- **table_css_clases** – class to be applied to the table.
- **table_id** – `str` for the id of the generated tabled.
- **template_name** – `str` template to render the table.
- **kwargs** – See below

Keyword Arguments

- **show_primary_key (bool) – Set if the**
  primary key of the model should be displayed, default=``False``.

- **added_columns (`list`) – Extra columns to show in the table,**
  should be a `list` object having the pair ((field_identifier, field_header), callable_function). Note that field_identifier is to mark a difference to the models fields and callable_function needs to be a function that will receive an object and return an str to print in the table column.
- **column_headers_css_classes** – CSS classes to be applied to the

column headers. Should be a dictionary having the fields as keys and the css classes to be applied as values.

`contains_field`(*lookups: list*, *field: str*) → int

Check if the field is in the sort lookups.

`get_sort_url`(*field: str*) → tuple[str, str, bool, bool]

Generate the urls to sort the table for the given field.

`get_table_body`() → str

Generate the body of the table.

`get_table_headers`() → str

Generate the column with the link to sort.

`render`() → str

Generate the table with the sort.

`sort_columns`(*field_order: list*)

Sort the columns according to the field order.

INDEX