django-sitetables Documentation

Release 0.2.1

Igor 'idle sign' Starikov

Feb 04, 2022

Contents

https://github.com/idlesign/django-sitetables

CHAPTER 1

Description

Reusable application for Django featuring DataTables integration

Offers:

- Various data sources support: models, query sets, list of dicts.
- `DataTables` plugins support: styling, internationalization, etc.
- Template tags for easy js and css inclusion.
- Template tags for DOM-based tables.

CHAPTER 2

Requirements

1. Python 3.6+
2. Django 1.10+

CHAPTER 3

Table of Contents

3.1 Quickstart

First place table definition into `views.py`:

```
from django.shortcuts import render
from sitetables.plugins.i18n import I18nPlugin
from sitetables.plugins.style.bootstrap4 import Bootstrap4Plugin
from sitetables.toolbox import Table

from .models import Entries


def entries(request):
    # We create client-side handled table from entries queryset,
    table_entries = Table(
        source=Entries.objects.filter(hidden=False),
        # We also activate some plugins.
        plugins=[
            I18nPlugin(),
            Bootstrap4Plugin(),
        ],
    )
    return render(request, 'entries.html', {'table_entries': table_entries})
```

Next create page template `entries.html`:

```
{% load sitetables %}

<!-- The following line usually goes into head tag. It'll load all needed css. -->
{% sitetables_css table_entries %}

<!-- The following resides in body tag. Note that in this scenario
     thead and tbody will be populated using JS automatically. -->
```

(continues on next page)

(continued from previous page)

```
<table id="table-entries" class="table table-striped table-condensed">
    <thead></thead><tbody></tbody>
</table>

<script type="text/javascript">
     $(function() {
         <!-- Initialize table using generated config.
              The following demonstrates how you can extend generated
              configuration. -->
         $('#table-entries').dataTable($.extend({},
             {% sitetable_config table_entries %},
             {
                 pagingType: 'full_numbers',
                 lengthChange: false,
             }
         ));

     });
 </script>

<!-- The following line usually goes somewhere near the end of the body.
     It'll load all needed js. -->
{% sitetables_js table_entries %}
```

3.1.1 Serverside tables

You can instruct `sitetables` to not to pour all table data to client, but to fetch it from server when needed. For that pass `on_server=True` to `Table`:

```
table_entries = Table(source=Entries, on_server=True)
```

3.1.2 Addressing tables by name

One may not spawn a table for each request, but create named table and address it by its name:

```
ENTRIES_TABLE_NAME = 'entries'
Table(source=Entries, name=ENTRIES_TABLE_NAME)

def entries(request):
    return render(request, 'entries.html', {'table_entries': ENTRIES_TABLE_NAME})
```

3.2 Plugins

Additional functionality and appearance for tables are available through plugins.

You can activate certain plugins using `plugins` argument for `Table`.

```
from sitetables.plugins.i18n import I18nPlugin
from sitetables.plugins.style.bootstrap4 import Bootstrap4Plugin
from sitetables.toolbox import Table
```

(continues on next page)

(continued from previous page)

```
table_entries = Table(
    source=source
    plugins=[
        # Let's activate a couple of plugins.
        I18nPlugin(),
        Bootstrap4Plugin(),
    ],
)
```

Include plugin assets (JS, CSS) on your pages using `{% sitetables_css table_entries %}` and `{% sitetables_js table_entries %}` template tags.

Note: Find plugins in `sitetables.plugins` package.

3.2.1 Styling (themes)

Theme plugins are available in `sitetables.plugins.style` package.