Nuit Documentation

Release 1.2.2

Ben Cardy

Feb 26, 2018

Contents

A front-end framework based upon Zurb Foundation for Django.

CHAPTER 1

Contents

1.1 Installation

1.1.1 Requirements

Nuit is built using SASS, and requires that you install the SASS binary in order to compile and compress the CSS. This won't be installed using `pip`, and will need to be installed manually (usually with `gem install sass`).

1.1.2 Installation

- Install Nuit with `pip`:

```
pip install django-nuit
```

- Add `nuit` to your `INSTALLED_APPS` in `settings.py`:

```
INSTALLED_APPS = (
    ...
    'nuit',
    ...
)
```

- Add the the following to the end of `settings.py`:

```
from django_autoconfig.autoconfig import configure_settings
configure_settings(globals())
```

- For production, add the following settings for django-pipeline (see their documentation for more detail):

```
PIPELINE_ENABLED = True
```

- Run `collectstatic`:

```
manage.py collectstatic
```

- You're now ready to start using Nuit in your templates and apps.

1.1.3 Dependencies

In case you're installing Nuit differently (e.g. from the git repo), make sure to install the following dependencies.

- django

  Fairly obvious - this is a Django library, afterall:

  ```
  pip install django
  ```

- django-pipeline

  In order to compile the CSS used by Nuit:

  ```
  pip install django-pipeline
  ```

- django-foundation-statics

  Nuit provides a suite to tools to help you easily build a consistent front-end interface for your Django application using Zurb's Foundation framework. This package contains the static files (CSS, Javscript) that makes up that framework:

  ```
  pip install django-foundation-statics
  ```

- django-foundation-icons

  This package contains Zurb's Foundation Icons static files:

  ```
  pip install django-foundation-icons
  ```

- django-bourbon

  This package contains SASS mixins used by Nuit:

  ```
  pip install django-bourbon
  ```

- django-autoconfig

  Allows you to include two lines of code in your `settings.py` that adds all the required Django settings for Nuit:

  ```
  pip install django-autoconfig
  ```

1.2 Quickstart

To get started with Nuit, follow the steps below. Note: this is a quick overview, and for a more in-depth explanation of each stage you should read the relevant documentation.

- Install Nuit following the *Installation* guidelines.
- In your app, define a base template that extends from *nuit/base.html*, that includes the relevant framework elements:

```
{% extend 'nuit/base.html' app_title='My application' topbar=True leftmenu=True %}
```

- Use this template to define everything you want on all pages in your application, such as the application-level menu:

```
{% block left_menu %}
    {% app_menu 'My Application' %}
        {% menu_item name='Page One' link='/page-one/' %}
        {% menu_item name='Page Two' link='/page-two/' %}
        {% menu_item name='Page Three' link='/page-three/' %}
        {% menu_item name='Page Four' link='/page-four/' %}
        {% menu_item name='Page Five' link='/page-five/' %}
    {% end_app_menu %}
{% endblock %}
```

- Inherit all other templates from this one:

```
{% extend 'my-base.html' %}
```

1.3 Customising with SASS

Because Nuit (and Foundation) are written with SASS, you can write your own SASS that can override the default Nuit/Foundation settings.

To do so, include a `nuit/_custom.scss` in the root of your static files directory. This will get imported by Nuit when loaded, and variables set in there will override Nuit's (and Foundation's) defaults.

1.3.1 Available Variables

For a list of most variables, please refer to the Foundation docs. The following variables are used by Nuit when compiling the SASS:

`$sidebar-bg`

Default `rgb(250, 250, 250)`

The color of the left and right menus.

`$sidebar-left-large-width`

Default `250px`

The with of the left menu on large screens.

`$sidebar-left-medium-width`

Default `200px`

The width of the left menu on medium screens.

`$sidebar-right-max-width`

Default `220px`

The maximum width of the right menu.

`$sidebar-gutter-spacing`

Default `10px`

The spacing between the menus and the content and window frame.

`$sidebar-link-active-color`

Default `$anchor-font-color`

The color of active links in the menus.

`$sidebar-link-hover-color`

Default `$anchor-font-color-hover`

The hover color of links in the menus.

`$sidebar-link-active-bg`

Default `rgba(0, 64, 84, 0.05)`

The background colour of active links in the menus.

`$sidebar-link-hover-bg`

Default `rgba(0, 64, 84, 0.03)`

The background colour of hovered links in the menus.

`$loader-colors`

Default `rgba(0, 0, 0, 1), rgba(0, 0, 0, 0.8), rgba(0, 0, 0, 0.6), rgba(0, 0, 0, 0.4), rgba(0, 0, 0, 0.2)`

The colours of the parts of the *Loading Animation*.

`$loader-subicon`

Default `"\0025CF"`

The icon of each part of the *Loading Animation*.

`$fixed-modal-width`

Default `400px`

The width of fixed-width modals, such as the login form and error pages.

1.3.2 Using Nuit's Styles

To use the variables that Nuit and Foundation themselves use in your own CSS, but without including any of their rendered CSS, add the following to the top of your SCSS file:

```
@import "nuit/_styles"
```

You'll then have access to the variables defined above, as well as any used by Foundation.

1.4 Settings

There are a handful of settings available to customise Nuit from `settings.py`.

`NUIT_GLOBAL_TITLE`

Defines the global title to be displayed in the *Top Bar*. Note that this is only visible if you do not define *`NUIT_LARGE_LOGO`* (and optionally *`NUIT_SMALL_LOGO`*). Defaults to 'Nuit'.

NUIT_GLOBAL_LINK
Defines the URL that the link in the top left of the *Top Bar* links to.

NUIT_LARGE_LOGO
Defines the path (to be passed to the `static` template tag) to the logo to be used in the title. Defaults to `None`, and no logo will be displayed.

NUIT_SMALL_LOGO
Defines the path (to be passed to the `static` template tag) to the logo to be used at small sizes. *NUIT_SMALL_LOGO* must also be defined - if set, this logo will replace the large logo at small screen sizes. Defaults to `None`.

NUIT_SEARCH_VIEW
Defines the view which the search box, if used, will submit to. If not set, the form will submit to `/search/`.

NUIT_SEARCH_PLACEHOLDER

Default `Search...`

Defines the placeholder text for the search box.

NUIT_SEARCH_PARAMETER

Default `q`

Defines the name of the parameter used for the search term when submitted.

NUIT_APPLICATIONS
Defines the global Nuit applications menu. This is intended to be the same across all of your sites using Nuit, and therefore ought to be added to `settings.py` via something like Puppet or Chef. These items are available in a drop-down menu on the right of the *Top Bar*. Multiple levels are acceptable. The structure should be a list of dictionaries with particular keys:

```
NUIT_APPLICATIONS = (
    {
        'name': 'One Link',
        'link': 'https://www.google.com/',
    },
    {
        'name': 'More under here',
        'subs': (
            {
                'name': 'Another Link',
                'link': 'https://pypi.python.org/',
            },
            {
                'name': 'Another Link 2',
                'link': 'https://www.ocado.com',
            },
        ),
    },
)
```

1.5 Templates

Nuit is primarily a front-end framework, and as such comes with a number of templates and template-related tools to provide you with a consistent user experience and lean templates.

1.5.1 Base Template

`nuit/base.html`

The base template contains the necessary markup to provide the basic framework elements and styling, and include the CSS and Javascript required for the Nuit interface to function. Most of the time, unless you're using one of the generic templates listed below, you'll be extending from `nuit/base.html` and overwriting the various blocks you require.

Arguments

The following arguments can be passed to the *extend* tag to change the behaviour of the base template:

`topbar`

Default `False`

Boolean that decides whether to display the *Top Bar* or not.

`leftmenu`

Default `False`

Boolean that decides whether to display the *Left Menu* or not.

`rightmenu`

Default `False`

Boolean that decides whether to display the *Right Menu* or not.

`app_title`

Default `None`

The name of the application. This is displayed in various places, such as the `<title>` element, and as a button on the top bar to take you back to the *`app_url`* defined.

`app_logo`

Default `None`

A more complex application title or logo that can contain HTML elements. This is displayed in a button on the top bar to take you back to the *`app_url`* defined, and overrides any *`app_title`* defined in this location. *`app_title`* is still used in the `<title>` element.

`app_url`

Default `None`

The base view of the application - this should be a url-reversable string.

`breadcrumbs`

Default `False`

Whether to display the breadcrumbs or not. See the *Breadcrumbs* documentation for more detail.

`topbar_classes`

Default `None`

A string containing extra classes to apply to the top bar for this particular page.

`show_user_info`

Default `False`

Boolean that decides whether to show the login information on the top right or not.

show_search

Default `False`

Boolean that decides whether to show the search box on the top right or not.

search_view

Default *NUIT_SEARCH_VIEW*

The view which the search form submits to.

search_placeholder

Default *NUIT_SEARCH_PLACEHOLDER*

The placeholder text for the search box.

search_parameter

Default *NUIT_SEARCH_PARAMETER*

The name of the search parameter.

Blocks

The following blocks can be overridden in subsequent templates to customise the content of the template - override these using the standard Django template model:

```
{% block block_name %}
This is my new content
{% endblock %}
```

title

Defines the title of the page, used in the `<title>` element.

content

The main content of the page. The space that this content fills will be determined by the *topbar*, *leftmenu* and *rightmenu* attributes.

head

Allows you to insert content into the `<head>` of the page.

css

Allows you to enter CSS `<link>` or `<style>` elements that will be inserted into the `<head>` element.

scripts

Allows you to enter Javascript `<script>` tags that will be included just before the closing `</body>` element.

top_bar_menu

Allows you to add links to the *Top Bar*. This is only visible when *`topbar`* is `True`. Intended to be used with the *menu_item* template tag.

top_bar_right_menu

Allows you to add links to the *Top Bar* on the right. This is only visible when *`topbar`* is `True`. Intended to be used with the *menu_item* template tag.

user_info_menu

Allows you to add links to the *Top Bar* in the user info dropdown. This is only visible when *`topbar`* and *`show_user_info`* are `True`. Intended to be used with the *menu_item* template tag.

left_menu

Allows you to add content to the *Left Menu*. This is only visible when *`leftmenu`* is `True`. Intended to be used with the *menu_section* template tag.

right_menu

Allows you to add links to the *Right Menu*. This is only visible when *`rightmenu`* is `True`. Intended to be used with the *menu_section* template tag.

breadcrumbs

Allows you to define breadcrumb links. See the *Breadcrumbs* documentation for more detail.

1.5.2 Components

There are three basic framework components provided by Nuit that can be enabled on a per-app or per-template basis. These are the top bar, left menu and right menu.

Top Bar

The top bar is Nuit's implementation of Foundation's Top Bar component. It provides a global application menu on the right hand side (defined in `settings.py` with the *`NUIT_APPLICATIONS`* variable), a logo and an optional number of links. This is enabled by extending from *nuit/base.html* and setting *`topbar`* to `True`.

By default, if you provide an *`app_title`* to the *extend* tag, the Top Bar will contain a link to the root of your application next to the logo on the left. To add additional menu items, define the *top_bar_menu* block and add some *menu_item* template tags:

```
{% extend 'nuit/base.html' topbar=True app_title='My App' %}
{% block top_bar_menu %}
    {% menu_item name='A Menu' link='http://www.google.com' %}
{% endblock %}
```

Left Menu

The left menu is intended for use as the main menu of an application, designed to alter on a per-application basis. It is situated on the left of the screen until the screen size drops to 'small' (mobile screens etc.), where the menu items are then accessible under the hamburger icon in the top bar. It's enabled by extending from *nuit/base.html* and setting `leftmenu` to `True`.

By default, this menu will be empty. There is a special template tag, *app_menu*, designed for use within the *left_menu* block, that builds the application menu for you, and adds an optional header. An example usage:

```
{% extend 'nuit/base.html' leftmenu=True %}
{% block left_menu %}
    {% app_menu "My App" %}
        {% menu_item name='A Menu' link='http://www.google.com' %}
    {% end_app_menu %}
{% endblock %}
```

You can also use the *menu_section* template tag to add additional sections to this menu, before or after the main application menu.

Right Menu

The right menu is intended for use as a page-specific menu. It floats on the right of the screen until the screen size drops to 'medium' (tablets, narrow windows etc.), where the items are then available under a gear icon in the top right. Links appear directly beneath the gear, whereas more detailed data structures appear in a modal dialog box with an activation link beneath the gear. It is eneabled by extending from *nuit/base.html* and setting `rightmenu` to `True`.

By default, this menu will be empty. It is designed for use with the *menu_section* template tag, to be populated with whatever content is necessary for each particular page by overriding the *right_menu* template block. In order for the medium (and under) drop-down gear menu to work properly, each *menu_section* needs to be given a unique ID per page. See the *menu_section* docs for more details.

Example usage:

```
{% extend 'nuit/base.html' rightmenu=True %}
{% block right_menu %}
    {% menu_section 'Links' is_list=True %}
        {% menu_item name='A Menu' link='http://www.google.com' %}
    {% end_menu_section %}
    {% menu_section link_name='Details' %}
        <p>Some details about this page</p>
    {% end_menu_section %}
{% endblock %}
```

Breadcrumbs

Nuit includes a block in the base template which you can fill with *menu_item* elements to create a breadcrumb trail on each page. The last breadcrumb in each trail is intended to indicate the current page, and is therefore styled as such.

The *breadcrumbs* block needs to be overriden, and will be displayed if your template extends from *nuit/base.html* and sets `breadcrumbs` to `True`. Remember, `{{block.super}}` may be needed if you use hierarchical templates.

Example usage:

```
{% block breadcrumbs %}
    {{block.super}}
    {% menu_item name='Publishers' link='/publishers/' %}
    {% menu_item name='New' link='/publishers/new/' %}
{% endblock %}
```

Loading Animation

Renders the HTML for a loading animation. Unlike the previous components, this can be included in any location in a template and uses Django's `include` syntax, rather than a template block. Takes two optional arguments:

`size`

Default 32

The size of the loader. There are a number built into Nuit: 16, 32, 48, 64, 80, and 96. This is applied by adding a class of `nuit-loader-sizeXX`, where XX is the defined size. You can use this in your own stylesheets to define your own size styles if necessary - see *Customising with SASS* for more details.

`loading_text`

Default None

Text to display with the loader. If not set, no text will be displayed.

To use, include the `nuit/includes/loader.html` template. Example usage:

```
{% include 'nuit/includes/loader.html' with size=64 loading_text='Loading...' %}
```

Customising with SASS

To build your own loader classes, use the following SASS mixin:

```
.nuit-loader-sizeMASSIVE {
  @include nuit-loader(1000px);
}
```

This can be imported from `nuit/_mixins.scss`:

```
@import "nuit/_mixins";
```

You can then reference the size in your templates:

```
{% include 'nuit/includes/loader.html' with size='MASSIVE' %}
```

Forms

Nuit provides a *foundation_form* template tag to enable complex rendering of HTML forms, for use with Django's Form classes. See the *Forms* documentation for more detail.

Messages

Nuit's *Base Template* templates will display any messages logged with Django's messaging framework at the top of the content area.

1.5.3 Forms

Foundation's form component is complex and provides many options. The *foundation_form* template tag enables you to render your Form objects with complete control over their layout.

Basic Usage

To use the template tag at it's most basic, don't provide any options between the open and closing tags, and pass your `Form` object:

```
<form>
{% foundation_form my_form_object %}
{% end_foundation_form %}
</form>
```

This will render all your specified form fields at full width. (In Foundation's grid terms, this means they are defined as `small-12`, `medium-12` and `large-12`.)

Note: Just like Django's own form rendering, the template tag will not render the opening and closing `<form>` elements, or the submit button. That's up to you.

CSRF

By default, the *foundation_form* template tag will insert the csrf_token required by `django.middleware.csrf.CsrfViewMiddleware`. If you're not using this middleware, or don't want the token itself, you can set *`csrf_enabled`* to `False` in the tag:

```
{% foundation_form my_form_object csrf_enabled=False %}
{% end_foundation_form %}
```

Collapsing the containing row

If you wish to use the form in the *Right Menu*, or somewhere else where you do not wish the extra side padding for the form elements, set *`collapse_container`* to `True`:

```
{% foundation_form my_form_object collapse_container=True %}
{% end_foundation_form %}
```

Layout Customisation

To customise which fields appear next to each other in the form, specify each row as a semi-colon-separated list between the form tags. For example, for a imaginary 'Customer' form:

```
{% foundation_form customer_form %}
    title; first_name; last_name
    email_address;
    password; confirm_password;
{% end_foundation_form %}
```

This will render the `title`, `first_name` and `last_name` fields at equal widths on one row, the `email_address` field on it's own row, and then the two `password` fields equal width on the final row. Again, in Foundation's grid terms, this means that the three first-row fields will have `small-12`, `medium-4` and `large-4`, the `email_address` is `small-12` etc., and the `password` fields are `small-12`, `medium-6` and `large-6` etc.

Any fields you do not specify will be appended to the end of the form at full width.

Note: No matter which fields you specify on each line, Nuit will default all fields to their own line for small screens (`small-12`). This can be overriden - see the *Custom Widths* section.

Custom Widths

If you don't want to rely on Nuit's automatic spacing of fields (for example, you know that the `title` field really out to be smaller than the `field_name` and `last_name` fields), you can provide a `dict` of options after each field containing values for either `small`, `medium` or `large` depending on what you want to override. For example, to make the `title` field take up 2 columns at medium-up, use the following markup:

```
{% foundation_form customer_form %}
    title {'medium': 2}; first_name; last_name
    email_address;
    password; confirm_password;
{% end_foundation_form %}
```

Nuit will automatically space out unspecified width fields as best it can to fill the remaining space.

Note: Just like Foundation, widths defined for smaller sizes will be inherited by larger sizes. In the example above, even though we only specify the `medium` value, the `title` field also take a width of `2` at `large` sizes. If you don't want this behaviour, then either set the larger sizes manually, or set the next size up to `0` to let Nuit calculate the spacings.

Pre- and Post-fix Labels

Foundation allows you to define pre- and post-fix labels that attach to form inputs. One example of such usage is prepending a `http://` before an input asking for a URL (note that this data is **not** included in the submitted data). Nuit lets you define this in the field's data `dict`:

```
{% foundation_form customer_form %}
    title; first_name; last_name
    email_address {'postfix': '@gmail.com'};
    password {'prefix': '6 chars or less'}; confirm_password;
{% end_foundation_form %}
```

Custom Widths

By default, each pre- or post-fix label will take up 3 of the 12 available grid widths at all sizes, and the field will take up 9. This can be customised in exactly the same way as for fields, detailed above in *Custom Widths*. Add options to the `dict` of `prefix_small`, `prefix_medium`, `prefix_large`, `postfix_small`, `postfix_medium` and `postfix_large`:

```
{% foundation_form customer_form %}
    title; first_name; last_name
    email_address {'postfix': '@gmail.com', 'postfix_medium': 4};
    password {'prefix': '6 chars or less'}; confirm_password;
{% end_foundation_form %}
```

The same rules for inheritance apply as in the *note* above.

Additional Field Options

The following options are also available in the field option `dict`:

`show_label`

Default `True`

Whether to display the field's label or not.

1.5.4 Generic Templates

Nuit's generic templates provide the front-end detail for common operations, such as listing objects. These tend to go hand-in-hand with Django's Generic Views.

`nuit/generic/list.html`

Displays a list of objects, linking to their detail URL, with options for pagination and filtering. This is designed to be used with Nuit's *`nuit.views.SearchableListView`*.

Arguments

The following arguments can be passed to the `extend` tag to change the behaviour of the list template:

`base`

Default 'nuit/base.html'

Defines the template which *nuit/generic/list.html* will extend. This defaults to *nuit/base.html*, but in most cases you will have made customisations (such as menu items etc.) to your own base template, and need this template to extend your own.

`search_verb`

Default 'Search'

Defines the verb to be displayed on the submit button of the search form, if applicable.

Blocks

The following blocks can be overridden in subsequent templates to customise the behaviour of the template. All blocks that are included in the template hierarchy are also available - by default, this means all blocks available in *nuit/base.html*.

Note: As always, the best way to override these blocks is to look at the source code for *nuit/generic/list.html*. This will give you a good idea about how Nuit implements parts of the template, and therefore a good indication of what is required if you wish to override the blocks.

title

Defines the title of the page, used in an `<h1>` element and in the `<title>`.

search_query

Displays above the list of objects when a search has been performed. Defaults to displaying the number of results and the term that was searched for.

object_list

Used to override the list of objects. By default, contains an unordered list of the object's `str` representations that links to their `get_absolute_url`.

`nuit/generic/400.html`

A generic template designed for display when an HTTP status code of `400` is returned. Used by the *`nuit.handlers.handler400()`* handler.

`nuit/generic/403.html`

A generic template designed for display when an HTTP status code of `403` is returned. Used by the *`nuit.handlers.handler403()`* handler.

`nuit/generic/404.html`

A generic template designed for display when an HTTP status code of `404` is returned. Used by the *`nuit.handlers.handler404()`* handler.

`nuit/generic/500.html`

A generic template designed for display when an HTTP status code of `500` is returned. Used by the *`nuit.handlers.handler500()`* handler.

`nuit/generic/login.html`

A template designed for use with Django's `django.contrib.auth.login` view. Use in your `urls.py` as such:

```
from django.conf.urls import patterns, include, url

urlpatterns = patterns('',
    url(r'^', include('myapp.urls')),
    url(r'^login/$', 'django.contrib.auth.views.login', {'template_name': 'nuit/
↪generic/login.html'}),
)
```

1.5.5 Templates Tags and Filters

Nuit provides a handlful of template tags and filters designed to abstract some of the complexities of writing templates for the framework.

Tags

`extend`

Extends another template, passing any keyword arguments to the parent template (prefixed by `nuit_`). For example:

```
{% extend 'nuit/base.html' topbar=True %}
```

results in the variable `nuit_topbar` being available (and set to `True`) in `nuit/base.html`.

`menu_section`

Intended for use inside one of either the *left_menu* or *right_menu* blocks. This block wraps any content in the markup required to render a section of the menu correctly.

The following arguments are optional:

`title`

Default None

The title of this section. If ommitted, no title will be displayed.

`link_name`

Default None

When used in the *right_menu* block, these sections appear under the gear at medium screens and smaller. This attribute defines the name of the link to be displayed in that menu which will activate the popup. This is required if *`title`* is not set. If ommitted, and *`title`* is set, then the title will be used.

`id`

Default None

A unique ID to use for the section. If ommitted, then the result of the *`link_name`* attribute calculation will be run through Django's slugify function, and that result used.

`is_list`

Default False

Whether this section contains a list of links or not. If `True`, then the correct HTML will be wrapped around the content to display the list in the right way. This is intended to be used with the *menu_item* template tag.

Example usage:

```
{% block right_menu %}

    {% menu_section "Details" link_name="Show some details" %}
        <p>These are some details.</p>
    {% end_menu_section %}

    {% menu_section "More stuff" %}
        <table>
            <tr>
                <td>A table</td>
            </tr>
        </table>
    {% end_menu_section %}

{% endblock %}
```

`menu_item`

Renders the HTML for a menu item. Intended for use between *menu_section* tags with `is_list` set to `True`. If the link resolves to a view that the user doesn't have permission to access, the link will not be displayed. Takes the following arguments:

`link`
The URL for the menu item. You can use a view identifier, which will be reversed if possible.:

```
{% menu_item link='name.of.my.view' name="My URL" %}
```

`name`
The display name of the link.

`always_display`
If you want the link to always be displayed, even if the user can't access it, set this to True.

`id`

Default slugified *`name`*

A unique ID for the link, intended for use by the *set_active_menu* tag. Is applied as a class to the `<li>` prepended with `menu-`.

`app_menu`

Renders the main application menu. Intended for use in the *right_menu* block, and should be filled with *menu_item* links. One optional argument:

`title`
The title of the application - displayed above the menu items.

Example usage:

```
{% block left_menu %}
    {% app_menu 'My Application' %}
```

```
        {% menu_item name='Page One' link='/page-one/' %}
        {% menu_item name='Page Two' link='/page-two/' %}
        {% menu_item name='Page Three' link='/page-three/' %}
        {% menu_item name='Page Four' link='/page-four/' %}
        {% menu_item name='Page Five' link='/page-five/' %}
    {% end_app_menu %}
{% endblock %}
```

set_active_menu

Renders an invisible element used by Javascript to set the `active` class on a menu item for this page. This must be within a template block that renders to the screen - it doesn't matter which one. It produces no visible output. The only required attribute is:

active_menu
The unique ID of the menu item to be activated. To highlight the menu item in the example above, the required code would be:

```
{% set_active_menu "my-url" %}
```

pagination_menu

Used internally by Nuit in the *nuit/generic/list.html* template to generate the pagination links for a paginated list of objects. The required attributes are:

page_obj
The Django Page object for the current view of data. This is provided by Django's ListView and Nuit's SearchableListView.

show_totals

Default True

Whether to show the total number of objects or not.

foundation_form

Renders a form using Foundation's form styles in the given layout. Takes one required arguement, and one optional:

form
The Django Form object. Works with all standard Django Form classes.

csrf_enabled

Default `True`

Whether to include the CSRF token or not.

collapse_container

Default `False`

Whether to collapse the side padding of the form elements.

Further options are specified between the *foundation_form* and `end_foundation_form` tags. For more details, see the *Forms* documentation. Example usage:

```
{% foundation_form my_form_object %}
    title, first_name, surname
    email_address
{% end_foundation_form %}
```

Filters

`message_class`

Given a message object from Django's `django.contrib.messages`, returns a string containing the CSS alert-box class for Foundation alert boxes corresponding to the message's level (`DEBUG`, `INFO`, `SUCCESS`, `WARNING` or `ERROR`). Used to style the message alert boxes:

```
{% for message in messages %}
    <div data-alert class='alert-box {{message|message_class}}'>{{message}}</div>
{% endfor %}
```

1.5.6 JavaScript

Nuit provides a number of JavaScript hooks to dynamically alter the Nuit elements of your page.

`nuit.`**`add_message`**(*alert_type*, *message*)

Inject a message box onto the page.

Parameters

- **`alert_type`** (*`str`*) – The type of message: one of `debug`, `info`, `success`, `warning`, or `error`.
- **`message`** (*`str`*) – The message to display.

`nuit.`**`confirmation_box`**(*options*)

Display a confirmation box.

Parameters **`options`** (*`dict`*) – Optional parameters to customise the confirmation box.

`title`

Default 'Confirmation required'

The title for the confirmation box.

`description`

Default '<p>Are you sure you wish to perform this action?</p>'

A more detailed description of what the confirmation box is for.

`size`

Default 'tiny'

A Foundation attribute for the size of the box: one of `tiny`, `small`, `medium`, `large`, `xlarge`. Applied as a class, so you can define your own.

`yes`

Default 'Yes'

The text for the confirm button.

no

Default 'No'

The text for the abort button.

on_confirm
A function that gets executed when the user clicks the confirm button.

on_abort
A function that gets executed when the user clicks the abort button.

`nuit.`**button_bar_value**(*button_bar*[, *value*])
Get or set the value of a button bar.

Parameters

- **button_bar** (*obj*) – a jQuery object of the button bar.
- **value** (*str*) – The value to set the object to. If ommitted, the current value is returned instead.

1.6 Views

Nuit provides a number of generic views for you to subclass.

class `nuit.views.`**SearchableListView**(***kwargs*)
Bases: `django.views.generic.list.ListView`

Render a list of objects that's searchable.

search_fields = ()
The fields the search will be performed on, using `queryset.filter`.

1.7 URL Handlers

Nuit proides the following URL handlers for consistent error pages across the site. See Django's `django.conf.urls` documentation for more details. Nuit Status Code Handlers

`nuit.handlers.`**generic_handler**(*request*, *template*, *status*, *context=None*)
Return a response with a particular status code, rendering a template with a specified context.

`nuit.handlers.`**handler400**(*request*)
View handling bad requests, to be used as `django.conf.urls.handler400`

`nuit.handlers.`**handler403**(*request*)
View handling permission denied exceptions, to be used as `django.conf.urls.handler403`

`nuit.handlers.`**handler404**(*request*)
View handling invalid URLs, to be used as `django.conf.urls.handler404`

`nuit.handlers.`**handler500**(*request*)
View handling execeptions, to be used as `django.conf.urls.handler500`

Python Module Index

Index