django-propeller Documentation

Release 1.4.1

Thorsten Froehlich

Mar 27, 2017

Contents:

CHAPTER 1

Installation

The preferred way to install `django-propeller` is `pip`:

```
$ pip install django-propeller
```

Alternatively, you can install download or clone this repo and install from its folder with:

```
$ pip install -e .
```

In your project, you should add `django-propeller` to your `requirements.txt`.

Be sure to use `virtualenv` if you develop python projects.

Add to INSTALLED_APPS in your `settings.py`:

```
    'django_propeller',
```

After installation, the *Quickstart* will get you on your way to using `django-propeller`.

CHAPTER 2

Quickstart

After *Installation*, you can use `django-propeller` in your templates.:

Load the `propeller` library and use the `propeller_*` tags:

Example template

```
{# Load the tag library #}
{% load propeller %}

{# Load CSS and JavaScript #}
{% propeller_css %}
{% propeller_javascript %}

{# Display django.contrib.messages as Bootstrap alerts #}
{% propeller_messages %}

{# Display a form #}
<form action="/url/to/submit/" method="post" class="form">
  {% csrf_token %}
  {% propeller_form form %}
  {% buttons %}
    <button type="submit" class="btn btn-primary">
      {% propeller_icon "star" %} Submit
    </button>
  {% endbuttons %}
</form>

{# Read the documentation for more information #}
```

Template tags and filters

Refer to *Template tags and filters* for more information.

Settings

You can set defaults for `django-propeller` in your settings file. Refer to *Settings* for more information.

Demo application

The demo application provides a number of useful examples.

clone the repo:

```
$ git clone https://github.com/tfroehlich82/django-propeller.git
```

cd into the cloned directory:

```
$ cd django-propeller
```

run the testserver:

```
$ python manage.py runserver
```

open your browser and browse to:

```
http://127.0.0.1:8000
```

CHAPTER 3

Template tags and filters

Note: All the following examples it is understood that you have already loaded the `propeller` template tag library, placing the code below in the beginning that each template that `propeller` template tag library will be used. Read the *Installation* and *Quickstart* sections on how to accomplish this.

propeller_form

`django_propeller.templatetags.propeller.`**`propeller_form`**`(`**args*, ***kwargs*`)`

Render a form

Tag name:

```
propeller_form
```

Parameters:

form The form that is to be rendered

exclude A list of field names (comma separated) that should not be rendered E.g. exclude=subject,bcc

See *propeller_field* for other arguments

Usage:

```
{% propeller_form form %}
```

Example:

```
{% propeller_form form layout='inline' %}
```

propeller_form_errors

django_propeller.templatetags.propeller.**propeller_form_errors**(**args*, ***kwargs*)

Render form errors

Tag name:

```
propeller_form_errors
```

Parameters:

form The form that is to be rendered

type Control which type of errors should be rendered.

One of the following values:

- `'all'`
- `'fields'`
- `'non_fields'`

default `'all'`

layout Context value that is available in the template `propeller/form_errors.html` as `layout`.

Usage:

```
{% propeller_form_errors form %}
```

Example:

```
{% propeller_form_errors form layout='inline' %}
```

propeller_formset

django_propeller.templatetags.propeller.**propeller_formset**(**args*, ***kwargs*)

Render a formset

Tag name:

```
propeller_formset
```

Parameters:

formset The formset that is being rendered

See *propeller_field* for other arguments

Usage:

```
{% propeller_formset formset %}
```

Example:

```
{% propeller_formset formset layout='horizontal' %}
```

propeller_formset_errors

django_propeller.templatetags.propeller.**propeller_formset_errors**(**args*, ***kwargs*)

Render formset errors

Tag name:

```
propeller_formset_errors
```

Parameters:

formset The formset that is being rendered

layout Context value that is available in the template `propeller/form_errors.html` as `layout`.

Usage:

```
{% propeller_formset_errors formset %}
```

Example:

```
{% propeller_formset_errors formset layout='inline' %}
```

propeller_field

django_propeller.templatetags.propeller.**propeller_field**(**args*, ***kwargs*)

Render a field

Tag name:

```
propeller_field
```

Parameters:

field The form field to be rendered

layout If set to `'horizontal'` then the field and label will be rendered side-by-side, as long as there is no `field_class` set as well.

form_group_class CSS class of the `div` that wraps the field and label.

default `'form-group'`

field_class CSS class of the `div` that wraps the field.

label_class CSS class of the `label` element. Will always have `control-label` as the last CSS class.

show_help Show the field's help text, if the field has help text.

default `True`

show_label Whether the show the label of the field.

default `True`

exclude A list of field names that should not be rendered

set_required When set to `True` and the field is required then the `required` attribute is set on the rendered field. This only works up to Django 1.8. Higher Django versions handle `required` natively.

default `True`

set_disabled When set to `True` then the `disabled` attribute is set on the rendered field. This only works up to Django 1.8. Higher Django versions handle `disabled` natively.

default `False`

size Controls the size of the rendered `div.form-group` through the use of CSS classes.

One of the following values:

- `'small'`
- `'medium'`
- `'large'`

placeholder Sets the placeholder text of a textbox

horizontal_label_class Class used on the label when the `layout` is set to `horizontal`.

default `'col-md-3'`. Can be changed in *Settings*

horizontal_field_class Class used on the field when the `layout` is set to `horizontal`.

default `'col-md-9'`. Can be changed in *Settings*

addon_before Text that should be prepended to the form field. See the *propeller docs <http://getpropeller.com/components/#input-groups-basic>* for an example.

addon_after Text that should be appended to the form field. See the *propeller docs <http://getpropeller.com/components/#input-groups-basic>* for an example.

addon_before_class Class used on the span when `addon_before` is used.

One of the following values:

- `'input-group-addon'`
- `'input-group-btn'`

default `input-group-addon`

addon_after_class Class used on the span when `addon_after` is used.

One of the following values:

- `'input-group-addon'`
- `'input-group-btn'`

default `input-group-addon`

error_css_class CSS class used when the field has an error

default `'has-error'`. Can be changed *Settings*

required_css_class CSS class used on the `div.form-group` to indicate a field is required

default `''`. Can be changed *Settings*

bound_css_class CSS class used when the field is bound

default `'has-success'`. Can be changed *Settings*

Usage:

```
{% propeller_field field %}
```

Example:

```
{% propeller_field field show_label=False %}
```

propeller_label

`django_propeller.templatetags.propeller.`**`propeller_label`**(**args*, ***kwargs*)

Render a label

Tag name:

```
propeller_label
```

Parameters:

content The label's text

label_for The value that will be in the `for` attribute of the rendered `<label>`

label_class The CSS class for the rendered `<label>`

label_title The value that will be in the `title` attribute of the rendered `<label>`

Usage:

```
{% propeller_label content %}
```

Example:

```
{% propeller_label "Email address" label_for="exampleInputEmail1" %}
```

propeller_button

`django_propeller.templatetags.propeller.`**`propeller_button`**(**args*, ***kwargs*)

Render a button

Tag name:

```
propeller_button
```

Parameters:

content The text to be displayed in the button

button_type Optional field defining what type of button this is.

Accepts one of the following values:

- `'submit'`
- `'reset'`

- `'button'`
- `'link'`

style Optional field defining which style button should have.

Accepts one of the following values:

- `'default'`
- `'raised'`
- `'flat'`
- `'outline'`

icon Name of an icon to render in the button's visible content. See *propeller_icon* for acceptable values.

button_class The class of button to use. If none is given, btn-default will be used.

extra_classes Any extra CSS classes that should be added to the button.

size Optional field to control the size of the button.

Accepts one of the following values:

- `'xs'`
- `'sm'`
- `'small'`
- `'md'`
- `'medium'`
- `'lg'`
- `'large'`

href Render the button as an `<a>` element. The `href` attribute is set with this value.

name Value of the `name` attribute of the rendered element.

value Value of the `value` attribute of the rendered element.

Usage:

```
{% propeller_button content %}
```

Example:

```
{% propeller_button "Save" button_type="submit" button_class="btn-primary" %}
```

propeller_icon

`django_propeller.templatetags.propeller.`**`propeller_icon`**(*icon*, ***kwargs*)

Render an icon

Tag name:

propeller_icon

Parameters:

icon Icon name. See the Propeller docs for all icons.

size Size of the icon. Must be one of 'xs', 'sm', 'md', or 'lg'. Default: 'sm'

extra_classes Extra CSS classes to add to the icon HTML. Optional

title A title for the icon (HTML title attrivute). Optional

Usage:

```
{% propeller_icon icon %}
```

Example:

```
{% propeller_icon "star" %}
```

propeller_bootstrap_icon

django_propeller.templatetags.propeller.**propeller_bootstrap_icon**(*icon*, ***kwargs*)

Render an icon

Tag name:

propeller_bootstrap_icon

Parameters:

icon Icon name. See the Bootstrap docs for all icons.

extra_classes Extra CSS classes to add to the icon HTML. Optional

title A title for the icon (HTML title attrivute). Optional

Usage:

```
{% propeller_bootstrap_icon icon %}
```

Example:

```
{% propeller_bootstrap_icon "star" %}
```

propeller_alert

django_propeller.templatetags.propeller.**propeller_alert**(*content*, *alert_type='info'*, *dismissable=True*)

Render an alert

Tag name:

```
propeller_alert
```

Parameters:

content HTML content of alert

alert_type

- `'info'`

- `'warning'`
- `'danger'`
- `'success'`

default `'info'`

dismissable boolean, is alert dismissable

default `True`

Usage:

```
{% propeller_alert content %}
```

Example:

```
{% propeller_alert "Something went wrong" alert_type='error' %}
```

buttons

`django_propeller.templatetags.propeller.`**`propeller_buttons`**(*parser*, *token*)

Render buttons for form

Tag name:

```
buttons
```

Parameters:

submit Text for a submit button

reset Text for a reset button

Usage:

```
{% buttons %}{% endbuttons %}
```

Example:

```
{% buttons submit='OK' reset="Cancel" %}{% endbuttons %}
```

`django_propeller.templatetags.propeller.`**`propeller_fab`**(**args*, ***kwargs*)

Render a floating action button

Tag name:

```
propeller_fab
```

Parameters:

content The text to be displayed in the button

button_type Optional field defining what type of button this is.

Accepts one of the following values:

- `'submit'`

- `'reset'`
- `'button'`
- `'link'`

style Optional field defining which style button should have.

Accepts one of the following values:

- `'default'`
- `'raised'`
- `'flat'`
- `'outline'`

icon Name of an icon to render in the button's visible content. See *propeller_icon* for acceptable values.

button_class The class of button to use. If none is given, btn-default will be used.

extra_classes Any extra CSS classes that should be added to the button.

size Optional field to control the size of the button.

Accepts one of the following values:

- `'xs'`
- `'sm'`
- `'small'`
- `'md'`
- `'medium'`
- `'lg'`
- `'large'`

href Render the button as an `<a>` element. The `href` attribute is set with this value.

name Value of the `name` attribute of the rendered element.

value Value of the `value` attribute of the rendered element.

Usage:

```
{% propeller_fab content %}
```

Example:

```
{% propeller_fab "Save" button_type="submit" button_class="btn-primary" %}
```

propeller_messages

`django_propeller.templatetags.propeller.`**`propeller_messages`**(*context*, **args*, ***kwargs*)

Show django.contrib.messages Messages in propeller alert containers.

In order to make the alerts dismissable (with the close button), we have to set the jquery parameter too when using the propeller_javascript tag.

Uses the template `propeller/messages.html`.

Tag name:

```
propeller_messages
```

Parameters:

None.

Usage:

```
{% propeller_messages %}
```

Example:

```
{% propeller_javascript jquery=1 %}
{% propeller_messages %}
```

propeller_pagination

`django_propeller.templatetags.propeller.`**`propeller_pagination`**(*page*, ***kwargs*)

Render pagination for a page

Tag name:

```
propeller_pagination
```

Parameters:

page The page of results to show.

pages_to_show Number of pages in total

default `11`

url URL to navigate to for pagination forward and pagination back.

default `None`

size Controls the size of the pagination through CSS. Defaults to being normal sized.

One of the following:

- `'small'`
- `'large'`

default `None`

extra Any extra page parameters.

default `None`

parameter_name Name of the paging URL parameter.

default `'page'`

Usage:

```
{% propeller_pagination page %}
```

Example:

```
{% propeller_pagination lines url="/pagination?page=1" size="large" %}
```

propeller_navbar

Hint: See also *Navbar* for how to create navbar instances

django_propeller.templatetags.propeller.**propeller_navbar**(*navbar*)

Render a navbar.

Tag name:

```
propeller_navbar
```

Parameters:

navbar The previously defined navbar instance

Usage:

```
{% propeller_navbar navbar_instance %}
```

propeller_card

Hint: See also *Card* for how to create card instances

django_propeller.templatetags.propeller.**propeller_card**(*card*)

Render a propeller card.

Tag name:

```
propeller_card
```

Parameters:

card The previously defined card instance

Usage:

```
{% propeller_card card_instance %}
```

propeller_textfilters

django_propeller.templatetags.propeller.**pmd_display_text**(*text*, *size=1*, *autoescape=True*)

Render text as a Propeller display text (heading).

Tag name:

```
pmd_display_text
```

Parameters:

```
size
    Controls the size of the heading.

    An integer from 1 (normal) to 4 (very large)

    :default: ``1``
```

Usage:

```
{{ text_variable|pmd_display_text:size }}
```

Example:

```
{{ my_text|pmd_display_text:3 }}
```

django_propeller.templatetags.propeller.**pmd_muted_text**(*text*, *autoescape=True*)
Render a muted text (secondary heading).

Tag name:

```
pmd_muted_text
```

Usage:

```
{{ text_variable|pmd_muted_text }}
```

django_propeller.templatetags.propeller.**pmd_lead_text**(*text*, *autoescape=True*)
Render text as a Propeller lead text (intro).

Tag name:

```
pmd_lead_text
```

Usage:

```
{{ text_variable|pmd_lead_text }}
```

django_propeller.templatetags.propeller.**pmd_mark_text**(*text*, *autoescape=True*)
Render highligthed text.

Tag name:

```
pmd_mark_text
```

Usage:

```
{{ text_variable|pmd_mark_text }}
```

django_propeller.templatetags.propeller.**pmd_strike_text**(*text*, *autoescape=True*)
Render striked text.

Tag name:

```
pmd_strike_text
```

Usage:

```
{{ text_variable|pmd_strike_text }}
```

django_propeller.templatetags.propeller.**pmd_underline_text**(*text*, *autoescape=True*)

Render underlined text.

Tag name:

```
pmd_underline_text
```

Usage:

```
{{ text_variable|pmd_underline_text }}
```

django_propeller.templatetags.propeller.**pmd_bold_text**(*text*, *autoescape=True*)

Render bold text.

Tag name:

```
pmd_bold_text
```

Usage:

```
{{ text_variable|pmd_bold_text }}
```

django_propeller.templatetags.propeller.**pmd_italic_text**(*text*, *autoescape=True*)

Render italic text.

Tag name:

```
pmd_italic_text
```

Usage:

```
{{ text_variable|pmd_italic_text }}
```

propeller_jquery_url

django_propeller.templatetags.propeller.**propeller_jquery_url**()

Return the full url to jQuery file to use

Default value: `//code.jquery.com/jquery.min.js`

This value is configurable, see Settings section

Tag name:

```
propeller_jquery_url
```

Usage:

```
{% propeller_jquery_url %}
```

Example:

```
{% propeller_jquery_url %}
```

propeller_javascript_url

django_propeller.templatetags.propeller.**propeller_javascript_url**()
Return the full url to the Propeller JavaScript library

Default value: `None`

This value is configurable, see Settings section

Tag name:

```
propeller_javascript_url
```

Usage:

```
{% propeller_javascript_url %}
```

Example:

```
{% propeller_javascript_url %}
```

propeller_css_url

django_propeller.templatetags.propeller.**propeller_css_url**()
Return the full url to the Propeller CSS library

Default value: `None`

This value is configurable, see Settings section

Tag name:

```
propeller_css_url
```

Usage:

```
{% propeller_css_url %}
```

Example:

```
{% propeller_css_url %}
```

propeller_css

django_propeller.templatetags.propeller.**propeller_css**()
Return HTML for Propeller CSS.

Adjust url in settings. If no url is returned, we don't want this statement to return any HTML. This is intended behavior.

Default value: `None`

This value is configurable, see Settings section

Tag name:

```
propeller_css
```

Usage:

```
{% propeller_css %}
```

Example:

```
{% propeller_css %}
```

propeller_javascript

`django_propeller.templatetags.propeller.`**`propeller_javascript`**(*jquery=None*)

Return HTML for Propeller JavaScript.

Adjust url in settings. If no url is returned, we don't want this statement to return any HTML. This is intended behavior.

Default value: `None`

This value is configurable, see Settings section

Tag name:

```
propeller_javascript
```

Parameters:

jquery Truthy to include jQuery as well as propeller

Usage:

```
{% propeller_javascript %}
```

Example:

```
{% propeller_javascript jquery=1 %}
```

CHAPTER 4

Navbar

NavBarLinkItem

class `django_propeller.navbar.`**`NavBarLinkItem`**(*name=''*, *url=None*, *icon=None*)
Generates a Link navbar item or a Link DropDown item.

Parameters:

name The display name for the item. (for example: 'Home')

url The address for the link item. Can be a absolute URL or a resolvable Django url. (for example: 'http://example.org' or 'home'). Optional.

icon not yet supported

NavBarDropDownDivider

class `django_propeller.navbar.`**`NavBarDropDownDivider`**
Generates a DropDown Divider item.

NavBarDropDownItem

class `django_propeller.navbar.`**`NavBarDropDownItem`**(*name=''*, *items=None*, *url=None*)
Generates a DropDown navbar item.

Parameters:

name The display name for the item. (for example: 'Home')

url The address for the link item. Can be a absolute URL or a resolvable Django url. (for example: 'http://example.org' or 'home'). Optional.

icon not yet supported

items A list containing NavBarLinkItems and/or NavBarDropDownDivider. Optional.

NavBar

class `django_propeller.navbar.`**`NavBar`**

NavBar is a class that generates a NavBar.

Parameters:

brandname The brand shown on the very left of the navbar.

brandurl The address for the brand name. Can be a absolute URL or a resolvable Django url. (for example: 'http://example.org' or 'home'). Optional.

items A list containing NavBarLinkItems and/or NavBarDropDownItems. Optional.

style_inverse Generate a dark navbar if true (default) or a light navbar if false.

style_static Sets the static style for the navbar. Static if true (default) or floating on top if false.

NavBar Example

navbar.py:

```
from django_propeller.navbar import NavBar, NavBarLinkItem, NavBarDropDownItem,␣
↪NavBarDropDownDivider

class DemoNavBar(NavBar):
    brandname = "Brand"
    brandurl = "http://example.org"
    items = [
        NavBarLinkItem("Link"),
        NavBarDropDownItem("DropDown", [
            NavBarLinkItem("Action"),
            NavBarLinkItem("Another action"),
            NavBarDropDownDivider(),
            NavBarLinkItem("Separated link"),
            NavBarDropDownDivider(),
            NavBarLinkItem("One more separated link"),
        ])

    ]
```

your_view.py:

```
from django.views.generic.base import TemplateView
from django_propeller.views import NavBarMixin
from your_project.navbar import DemoNavBar

class HomePageView(TemplateView, NavBarMixin):
    template_name = 'your_template.html'
    navbar_class = DemoNavBar
```

your_template.html:

```
{% load propeller %}
{% propeller_navbar navbar %}
```

CHAPTER 5

Card

CardTitle

class django_propeller.card.**CardTitle**
Renders a Card Title.

Parameters:

text The display text for the title.

size The size for the title as integer. Works with the h-tag, so size=1 is bigger than size=3. Optional. (default=3)

CardSubtitle

class django_propeller.card.**CardSubtitle**
Renders a Card Subtitle.

Parameters:

text The display text for the subtitle.

CardBody

class django_propeller.card.**CardBody**
Renders a Card Body.

Parameters:

text The text to display in the body.

CardHeader

class django_propeller.card.**CardHeader**
Renders a Card Header.

Parameters:

content_left A list of items to display on the left of header. May contain Button, FAB, Image, CardTitle, or CardSubtitle.

content_middle A list of items to display in the middle of header. May contain Button, FAB, Image, CardTitle, or CardSubtitle.

CardMediaActions

class django_propeller.card.**CardMediaActions**
Renders Card Media Actions.

Parameters:

items A list of items to display in the Card Media Action section. May contain Button or FAB.

CardActions

class django_propeller.card.**CardActions**
Renders Card Actions.

Parameters:

items A list of items to display in the Card Action section. May contain Button or FAB.

CardMediaImage

class django_propeller.card.**CardMediaImage**
Renders a Card Media Image.

Parameters:

image Must be an instance of an Image.

CardMedia

class django_propeller.card.**CardMedia**
Renders Card Media

Parameters:

content

if style_inline=True: A list of items to display in the card media section. May contain CardMediaImage, CardTitle, or CardSubtitle.

or if style_inline=False: (default) A instance of CardMediaImage

style_inline Display card with inline style if true. (Default: False)

Card

class `django_propeller.card.`**`Card`**

Card is a class that generates a Propeller Card.

Parameters:

primary_title An instance of CardTitle. Optional.

secondary_title An instance of CardSubtitle. Optional.

header An instance of CardHeader. Optional.

media An instance of CardMedia. Optional.

body An instance of CardBody. Optional.

actions An instance of CardActions. Optional.

media_actions An instance of CardMediaActions. Optional.

style_inverse Display dark style if true. (Default: False)

style_inline Display card with inline style if true. (Default: False)

width Width of the card in Bootstrap grid (col-md) as integer. (Default: 4)

Card Example

card.py:

```
from django_propeller.card import Card, CardHeader, CardActions, CardMediaActions,␣
↪CardTitle, CardSubtitle, \
CardMediaImage, CardBody, CardMedia
from django_propeller.components import Image, Button, FAB

class DemoTitle(CardTitle):
    text = "Title goes here"
    size = 2


class DemoSubtitle(CardSubtitle):
    text = "Secondary text"


class DemoMediaImage(CardMediaImage):
    image = Image(source="http://propeller.in/assets/images/profile-pic.png",␣
↪responsive=True)


class DemoMedia(CardMedia):
    content = [DemoMediaImage(), ]


class DemoBody(CardBody):
    text = "Cards provide context and an entry point to more robust information and␣
↪views. " \
```

```
            "Don't overload cards with extraneous information or actions."


class DemoHeader(CardHeader):
    content_middle = [DemoTitle(), DemoSubtitle()]


class DemoActions(CardActions):
    items = [
        Button(content='primary', button_class='btn-primary'),
        Button('Action'),
    ]


class DemoMediaActions(CardMediaActions):
    items = [
        FAB('', button_class='btn-primary', icon='share', style='flat', size='sm'),
        FAB('', button_class='btn-primary', icon='thumb_up', style='flat', size='sm'),
        FAB('', button_class='btn-primary', icon='drafts', style='flat', size='sm')
    ]


class DemoCard(Card):
    primary_title = DemoTitle()
    secondary_title = DemoSubtitle()
    header = DemoHeader()
    actions = DemoActions()
    media_actions = DemoMediaActions()
    media = DemoMedia()
    body = DemoBody()
```

your_template.html:

```
{% load propeller %}
{% propeller_card card %}
```

CHAPTER 6

Settings

The django-propeller has some pre-configured settings.

They can be modified by adding a dict variable called `PROPELLER` in your `settings.py` and customizing the values you want;

The `PROPELLER` dict variable contains these settings and defaults:

```
# Default settings
PROPELLER = {

    # The URL to the jQuery JavaScript file
    'jquery_url': '//code.jquery.com/jquery.min.js',

    # The Bootstrap base URL
    'base_url': '//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/',

    # The complete URL to the Bootstrap CSS file (None means derive it from base_url)
    'css_url': None,

    # The complete URL to the Bootstrap CSS file (None means no theme)
    'theme_url': None,

    # The complete URL to the Bootstrap JavaScript file (None means derive it from␣
↪base_url)
    'javascript_url': None,

    # Put JavaScript in the HEAD section of the HTML document (only relevant if you␣
↪use propeller.html)
    'javascript_in_head': False,

    # Include jQuery with Bootstrap JavaScript (affects django-propeller template␣
↪tags)
    'include_jquery': False,

    # Label class to use in horizontal forms
    'horizontal_label_class': 'col-md-3',
```

```
    # Field class to use in horizontal forms
    'horizontal_field_class': 'col-md-9',

    # Set HTML required attribute on required fields, for Django <= 1.8 only
    'set_required': True,

    # Set HTML disabled attribute on disabled fields, for Django <= 1.8 only
    'set_disabled': False,

    # Set placeholder attributes to label if no placeholder is provided
    'set_placeholder': True,

    # Class to indicate required (better to set this in your Django form)
    'required_css_class': '',

    # Class to indicate error (better to set this in your Django form)
    'error_css_class': 'has-error',

    # Class to indicate success, meaning the field has valid input (better to set␣
↪this in your Django form)
    'success_css_class': 'has-success',

    # Renderers (only set these if you have studied the source and understand the␣
↪inner workings)
    'formset_renderers':{
        'default': 'propeller.renderers.FormsetRenderer',
    },
    'form_renderers': {
        'default': 'propeller.renderers.FormRenderer',
    },
    'field_renderers': {
        'default': 'propeller.renderers.FieldRenderer',
        'inline': 'propeller.renderers.InlineFieldRenderer',
    },
}
```

CHAPTER 7

Templates

You can customize the output of `django-propeller` by writing your own templates. These templates are available:

propeller/field_help_text_and_errors.html

This renders the help text and error of each field.

Variable `help_text_and_errors` contains an array of strings.

propeller/form_errors.html

This renders the non field errors of a form.

Variable `errors` contains an array of strings.

propeller/messages.html

This renders the Django messages variable.

Variable `messages` contains the messages as described in https://docs.djangoproject.com/en/dev/ref/contrib/messages/#displaying-messages

`messages` is passed through three built-in filters

safe <https://docs.djangoproject.com/en/dev/ref/templates/builtins/#std:templatefilter-safe>

urlize <https://docs.djangoproject.com/en/dev/ref/templates/builtins/#std:templatefilter-urlize>

linebreaksbr <https://docs.djangoproject.com/en/dev/ref/templates/builtins/#std:templatefilter-linebreaksbr>

Other

There are two more templates, `propeller/propeller.html` and `propeller/pagination.html`. You should consider these private for now, meaning you can use them but not modify them.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index