# django-cruditor

unknown

Feb 26, 2023

# CONTENTS


**1 All Contents** **1**
- 1.1 Installation . . . 1
- 1.2 Usage . . . 1
- 1.3 API Reference . . . 2
- 1.4 Changelog . . . 10

**2 Indices and tables** **15**

**Python Module Index** **17**

**Index** **19**

CHAPTER

ONE

# ALL CONTENTS

## 1.1 Installation

django-cruditor supports Python 3 only and requires at least Django 1.11 (because of the template based widget rendering). Optional dependencies are django-tables2, django-filter and django-tapeforms. Depending on what parts of django-cruditor you want to use, you have to install them manually.

To start, simply install the latest stable package using the command

```
$ pip install django-cruditor
```

In addition, you have to add `'cruditor'` to the `INSTALLED_APPS` setting in your `settings.py`.

If you're planing to us the ListViews provided django-cruditor, you have to install `django-tables2` and add need to add `'django_tables2'` to your `INSTALLED_APPS`.

If you want to use the form rendering of the Add- or ChangeViews, you'll need `django-tapeforms`. Don't forget to add `tapeforms` to your settings' `INSTALLED_APPS`.

Finally, if you want to use the filtering capabilities of django-cruditor ListViews, install `django-filter` and add `django_filters` to your `INSTALLED_APPS`.

Thats it, now continue to the *Usage section* to learn how to render your forms to HTML.

## 1.2 Usage

The best you can do right now is have a look on the examples. Every feature is demo'ed there.

- Minimal code one has to write to create a view in the Cruditor context (examples/minimal)
- How to provide a classic List/Add/Change/Delete set of views with nearly no effort (examples/collection)
- How to provide filters in the list view to find the right objects (examples/collection)
- How to work with data not coming form the Django ORM (examples/remote)
- How to integrate inline formsets for related data (examples/formset)

To start, just check out the repository and run the examples project. After creating a superuser you can log in and find all demos in the menu.

## 1.3 API Reference

### 1.3.1 Mixins

**class** cruditor.mixins.**CruditorMixin**

Bases: object

Base mixin for all Cruditor views. Provides common functionality for all views.

It is a good idea to have a own "base" mixin to configure the common options like `menu_title`, the urls and templates.

Usually you might have `required_permission` configured per-view.

**menu_title = 'CRUDitor'**

Title to use in templates / menu bar

**index_url = '#'**

URL to use in the linked menu_title

**logout_url = '#'**

URL to use when providing a logout link to the user.

**change_password_url = None**

URL to the change password view, if available.

**menu_template_name = 'cruditor/includes/menu.html'**

Template name which is included to render the menu.

**extrahead_template_name = 'cruditor/includes/extrahead.html'**

Template used to include extra head stuff.

**login_template_name = 'cruditor/login.html'**

Page template for the login view.

**login_form_class**

Form class which is used in the login view.

alias of *LoginForm*

**staff_required = True**

Decide if only staff users should be able to use the Cruditor views.

**required_permission = None**

Which permission is required to access the view.

**model_verbose_name = None**

If not provided, Cruditor tries to look up the verbose name from `model.Meta`

**dispatch**(*request*, **args*, ***kwargs*)

Ensure the user is logged in (by calling *ensure_logged_in* ` method). If the user is logged in, permissions are checked by calling `ensure_required_permission`.

**get_cruditor_context**(*alternative_title=None*, *login_context=False*)

Provides some context for all Cruditor templates to render menu, header, breadcrumb and title buttons.

The method takes an optional argument `alternative_title` to override the default title from `get_title` method.

**get_title**()

Returns the title of the page. Uses view's `title` property. If not set falls back to `menu_title`.

**get_breadcrumb_title**()

By default, the breadcrumb title is the same as the page title. Calls `get_title` if not overwritten.

**get_breadcrumb**()

This method is expected to return a list of breadcrumb elements as a list.

Every breadcrumb element is a dict or object with at least a `title` property/key. If a `url` key/property is provided, the item is linked.

**get_titlebuttons**()

This method is expected to return None or a list of buttons to display in the title row of the page.

Every button element is a dict or object with at least a `label` and `url` property/key. In addition, one can provide an alternative `button_class` which is used as a css class - pefixed with "btn-". Default `button_class` is "light".

**get_model_verbose_name**()

Returns the verbose name of the handled object/item.

If `model_verbose_name` is set, the value is used. If not, Cruditor tries to get the verbose name from the model property (via Meta class). If no name is available at all, "Item" is returned.

**ensure_logged_in**(*request*, **args*, ***kwargs*)

This method checks if the request user is logged in and has the right flags set (e.g. `is_staff` if `staff_required` is set in view).

If user is logged in, `True` is returned. If not, `handle_not_logged_in` is called.

**handle_not_logged_in**(*request*, **args*, ***kwargs*)

This method is responsible to handle not logged-in users. By default, renders the Django login view using a Cruditor optimized template using the `login_form_class` as Form.

**get_required_permission**()

Returns the required Django permissions required to access the view.

You might override the method to apply more complex rules on what permissions are required.

**ensure_required_permission**()

This method ensures that all required permissions (fetched by calling `get_required_permission`).

If permissions are not met, `PermissionDenied` is raised.

**get_context_data**(***kwargs*)

Adds the `cruditor` context variable to the template context. Uses data from `get_cruditor_context` method.

**class** cruditor.mixins.**FormViewMixin**

Bases: `object`

Mixin to add formset support to Django FormViews. To use formsets, you have to provide a set of formsets as a dict (or OrderedDict if you have more than one formset - just to have a defined ordering).

**formset_classes = None**

**get_formset_classes**()

This method returns the formset classes to render in the form view. By default, returns the `formset_classes` property.

**get_formset_kwargs**(*formset_class*)

This method returns additional kwargs to initialize a formset. The *formset_class* is provided to ensure the method can return proper kwargs.

**get**(*request*, **args*, ***kwargs*)

Extended get-method to render to form and all formsets properly initialized.

**post**(*request*, **args*, ***kwargs*)

Extended version of the FormView.post method which validates the form and all configured formsets. If everything is valid, `form_valid` is called. If something is not valid, `form_invalid` is called.

Both the form instance and all formset instances are provided to the called method. The form is passed as the first argument, the formsets are passed as keyword arguments using the formset key from `formset_classes`.

**save_form**(*form*, ***formsets*)

This method is called from `form_valid` to actual save the data from the form and all formsets. All saving is done by default.

**get_success_message**()

Returns the success message to display when the form is valid.

**form_valid**(*form*, ***formsets*)

Saves the data and provides a nice success message, then redirects to the `get_success_url` url.

**form_invalid**(*form*, ***formsets*)

Re-render the page with the invalid form and/or formsets.

### 1.3.2 Forms

**class** cruditor.forms.**CruditorTapeformMixin**(**args*, ***kwargs*)

Bases: `Bootstrap4TapeformMixin`

Cruditor mixin for all forms (relies on django-tapeforms).

**class** cruditor.forms.**CruditorFormsetMixin**

Bases: `object`

Helper mixin to provide some additional configuration to the javascript part of Cruditor's formset support, mainly translations but also all other stuff which might be needed.

**js_formset_options = None**

**template_context**

This cached property can be used to access extra context in the formset template while keeping the performance up by not "generating" the context over and over again. The return value of this property is generated by calling the method *get_template_context* once per formset instance.

**get_template_context**()

This method builds a context which is used by the *template_context* property to return additional context when rendering the formset. Some defaults are already set and might be overwritten.

**add_fields**(*form*, *index*)

Overwritten method to make sure the DELETE marker field is hidden in output.

**get_js_formset_options**()

This method builds the options dict for the javascript part. Some defaults are merged with *js_formset_options* property.

**class** cruditor.forms.**CruditorFormsetFormMixin**(**args*, ***kwargs*)

Bases: *CruditorTapeformMixin*

Helper mixin for forms in a formset, used together with Cruditor-enabled formsets.

**visible_fields**()

This method is overwritten to make sure that the DELETE marker field is not considered when returning the list of visible fields.

**hidden_fields**()

This method is overwritten to make sure that the DELETE marker field is not considered when returning the list of hidden fields. Cruditor template renders the field manually.

**class** cruditor.forms.**LoginForm**(**args*, ***kwargs*)

Bases: *CruditorTapeformMixin*, AuthenticationForm

Tapeform-enabled version of the Django AuthenticationForm.

**base_fields = {'password': <django.forms.fields.CharField object>, 'username': <django.contrib.auth.forms.UsernameField object>}**

**declared_fields = {'password': <django.forms.fields.CharField object>, 'username': <django.contrib.auth.forms.UsernameField object>}**

**property media**

Return all media required to render the widgets on this form.

**class** cruditor.forms.**ChangePasswordForm**(**args*, ***kwargs*)

Bases: *CruditorTapeformMixin*, SetPasswordForm

Tapeform-enabled version of the Django SetPasswordForm.

**base_fields = {'new_password1': <django.forms.fields.CharField object>, 'new_password2': <django.forms.fields.CharField object>}**

**declared_fields = {'new_password1': <django.forms.fields.CharField object>, 'new_password2': <django.forms.fields.CharField object>}**

**property media**

Return all media required to render the widgets on this form.

### 1.3.3 Filters

**class** cruditor.filters.**AnyChoiceFilter**(**args*, ***kwargs*)

Bases: ChoiceFilter

Extended ChoiceFilter which adds an "any" choice to the choices from the field / provided options by setting a empty_label on the generated form field.

**class** cruditor.filters.**MultiCharFilter**(*fields*, **args*, ***kwargs*)

Bases: CharFilter

This filter performs an OR query on the defined fields from a single entered value.

The following will work similar to the default UserAdmin search:

```
class UserFilterSet(FilterSet):
    search = MultiCharFilter([
        'username', 'first_name', 'last_name', '^email'])

    class Meta:
        model = User
        fields = ['search']
```

The filter supports filtering in different modes (icontains, istartswith, iexact, and search). icontains is the default mode, use ^, = and @ in the list of fields for the other modes.

Based on some ideas from https://gist.github.com/nkryptic/4727865

**default_lookup_type = 'icontains'**

**lookup_types = [('^', 'istartswith'), ('=', 'iexact'), ('@', 'search')]**

**filter**(*qs*, *value*)

### 1.3.4 Views

**class** cruditor.views.**Cruditor404View**(***kwargs*)

Bases: *CruditorMixin*, TemplateView

Customized not found page. Needed to add the required cruditor context for properly rendered templates.

**template_name = 'cruditor/404.html'**

Template used to render the 404 page.

**dispatch**(*request*, **args*, ***kwargs*)

Ensure the user is logged in (by calling *ensure_logged_in* ` method). If the user is logged in, permissions are checked by calling `ensure_required_permission`.

**class** cruditor.views.**Cruditor403View**(***kwargs*)

Bases: *CruditorMixin*, TemplateView

Customized permission denied page. Needed to add the required cruditor context for properly rendered templates.

**template_name = 'cruditor/403.html'**

Template used to render the 403 page.

**dispatch**(*request*, **args*, ***kwargs*)

Ensure the user is logged in (by calling *ensure_logged_in* ` method). If the user is logged in, permissions are checked by calling `ensure_required_permission`.

**class** cruditor.views.**CruditorListView**(***kwargs*)

Bases: *CruditorMixin*, TemplateView

Enhanced list view backed by django-tables2 and django-filters.

You want to set at least the `model` property. The remaining default property values are just fine for a working output.

By providing a alternative `table_class` and/or `filter_class` you can improve the usability of the view even further.

**model = None**

Model to work on in this view.

**queryset = None**

Queryset to use for looking up objects.

**filter_class = None**

Optional django_filters.FilterSet class to provide filtering capabilities.

**table_class = None**

Required django_tables2.Table class to change the rendered table.

**template_name = 'cruditor/list.html'**

Template to use when rendering the list view.

**get_context_data**(***kwargs*)

Prepares the context by adding the `table` context variable. If you have configured `filter_class`, the `filter_form` context variable will be provided too.

**get_queryset**()

Provide a queryset to fetch data with. If `queryset` is set on the class, the value will be used, if not but `model` is set, the default manager is used.

If both queryset and model is not set, you have to override this method to provide data to display.

**get_table_class**()

Method to override the used table class. By default, returns `table_class` property if set.

If no `table_class``is provided, an ``ImproperlyConfigured` exception is raised.

**get_table_kwargs**()

Override to provide additional kwargs when initializing the table object.

**get_filter_class**()

Method to override the used filter class from django_filters. By default, returns `filter_class` property if set.

**get_filter_kwargs**()

Override to provide additional kwargs when initializing the filterset object.

**get_filtered_queryset**()

Filter the base queryset using the django-filters FilterSet if available. QuerySet is passed if no `filter_class` is defined.

**get_table**(*filtered_qs*)

Prepare the table object using the provided QuerySet/Iterable.

**class** cruditor.views.**CruditorAddView**(***kwargs*)

Bases: *CruditorMixin*, *FormViewMixin*, CreateView

Enhanced view to add new items using a form view.

**success_message = 'The {model} "{object}" was successfully added.'**

Message used when a new item was added successfully.

**template_name = 'cruditor/form.html'**

Template used to render the add form view.

**get_object**()

As we are in a add view, no object will be available ever.

**get_title**()

Generate a sane title when adding new items using the `get_model_verbose_name`.

**class** cruditor.views.**CruditorChangeView**(***kwargs*)

Bases: *CruditorMixin*, *FormViewMixin*, UpdateView

Enhanced view to edit existing items using a form view.

**success_message = 'The {model} "{object}" was successfully changed.'**

Message used when a item was changed successfully.

**template_name = 'cruditor/form.html'**

Template used to render the change form view.

**get_title**()

Generate a sane title when editing an item using the __str__ representation of a object.

**get_delete_url**()

Override to provide a link for the delete button in the change view. By default no delete button is visible.

**get_context_data**(***kwargs*)

Add the `object_delete_url` context variable using `get_delete_url`. Feel free to extend the context further.

**class** cruditor.views.**CruditorDeleteView**(**args*, ***kwargs*)

Bases: *CruditorMixin*, DeleteView

Enhanced view to delete existing items after a confirmation.

**success_message = 'The {model} "{object}" was successfully deleted.'**

Message used when a item was deleted.

**template_name = 'cruditor/delete.html'**

Template used to render the confirmation form view.

**delete**(**args*, ***kwargs*)

Call the delete() method on the fetched object and then redirect to the success URL.

**form_valid**(*request*, **args*, ***kwargs*)

Call `perform_delete` method and redirect to the success URL with a nice success message. If there are protected related objects, an error message is shown instead with the output of `format_linked_objects`.

**get_title**()

Generate a sane title when requesting a confirmation to delete an item using the __str__ representation of a object.

**perform_delete**()

Actual delete a object/model/item after confirmation.

**format_linked_objects**(*objects*)

Generate a list of strings describing the objects which have a protected relation to the item to delete.

**class** cruditor.views.**CruditorChangePasswordView**(***kwargs*)

Bases: *CruditorMixin*, FormView

Enhanced view to perform password changes in the Cruditor context.

**template_name = 'cruditor/form.html'**

Template used when rendering the change password form.

**title = 'Change password'**

Title for breadcrumb and page.

**form_class**

Form used to change the password.

alias of *ChangePasswordForm*

**get_form_kwargs**()

The current user is passed to the provided `form_class` when initializing the change password form.

**form_valid**(*form*)

Save the new password (by calling `form.save`) and rotate the session authorization hash.

**get_context_data**(***kwargs*)

Set `form_save_button_label` context variable to change the button label for the change password form.

**class** cruditor.views.**CruditorLogoutView**(***kwargs*)

Bases: *CruditorMixin*, LogoutView

View to log out the current user. After logging out, a info is displayed.

**template_name = 'cruditor/logout.html'**

Template used to display the info that the user was logged out.

**ensure_logged_in**(**args*, ***kwargs*)

This method checks if the request user is logged in and has the right flags set (e.g. `is_staff` if `staff_required` is set in view).

If user is logged in, `True` is returned. If not, `handle_not_logged_in` is called.

**get_context_data**(***kwargs*)

Adds the `cruditor` context variable to the template context. Uses data from `get_cruditor_context` method.

### 1.3.5 Collection

**class** cruditor.collection.**CollectionViewMixin**

Bases: object

Mixin to provide some extra default functionality to Cruditor views to make building views for a collection of data (like a Django model) even easier.

**collection_list_title = 'Collection'**

Title for collection list view

**collection_list_urlname = None**

URL name to use when linking to the list view (e.g. in breadcrumb)

**collection_detail_urlname = None**

URL name when linking to a detail page of a item (e.g. in list table or breadcrumb)

**get_title**()

The method calls the `get_collection_list_title` method when the view is a list view. All other views rely on the default behavior of cruditor.

`get_breadcrumb_title()`

The breadcrumb title returns "Delete" for delete views, default breadcrumb for all other views.

`get_breadcrumb()`

This method creates the required breadcrumb items for the collection following some rules:

- No extra items for list view
- list view element when the `collection_include_list_crumb` method is true.
- detail view element when the `collection_include_detail_crumb` method is true.

`get_table_class()`

This method returns the django-tables2 Table class to use in the list view. If no class is defined, a new Table class is created (with one linked column).

`collection_include_list_crumb()`

If this method returns true, the list view should be included in the breadcrumb.

`collection_include_detail_crumb()`

If this method returns true, the detail view should be included in the breadcrumb.

`get_collection_list_title()`

Helper method to override the used collection list title. By default, just returns the class `collection_list_title` property.

`get_collection_list_url()`

Helper method to generate the collection list url. By default, just calls reverse with the `collection_list_urlname` property.

`get_collection_detail_title()`

Helper method to override the used collection detail title. By default, just returns the str-representation of the requested item.

`get_collection_detail_url()`

Helper method to generate the collection detail url for the current object. By default, calls reverse with the `collection_detail_urlname` property and passes the object pk to the function call.

## 1.4 Changelog

### 1.4.1 2.4.0 - 2023-02-26

- Add support for Django 4.1

### 1.4.2 2.3.3 - 2022-09-19

- Add more blocks to form template.

### 1.4.3 2.3.2 - 2022-09-08

- Expose newForm in formset add callback.
- Add more template blocks for formset template.

### 1.4.4 2.3.1 - 2022-08-04

- Fix issue with logout view

### 1.4.5 2.3.0 - 2022-08-03

- Fix issue with small buttons
- Fix spacing issue with title buttons
- Add support for target in title buttons.

### 1.4.6 2.2.0 - 2022-08-02

- Ship scss and js files with source distribution
- Add extra blocks for title section for better customization

### 1.4.7 2.1.0 - 2022-06-23

- Add get_formset_kwargs method to FormViewMixin

### 1.4.8 2.0.0 - 2022-05-09

- Drop support for Django < 2.2
- Add template_context property for formsets

### 1.4.9 1.4.0 - 2019-09-26

- Add method to customize the success message of change form views

### 1.4.10 1.3.2 - 2019-08-16

- Improve formset js component

### 1.4.11 1.3.1 - 2019-08-16

- Fix issue with formsets and duplicated DELETE inputs
- Fix broken delete callback in formset js module

### 1.4.12 1.3.0 - 2019-07-08

- Catch the exception and show an error message when deleting an item with protected related objects
- Remove `DeleteConfirmForm` to replace the checkbox by a simple message and make the deletion process lighter

### 1.4.13 1.2.1 - 2019-03-26

- Fix a bug when a user is not logged in but get_titlebuttons/get_breadcrumb relies on self.object

### 1.4.14 1.2.0 - 2019-03-18

- Add French translations
- Improve templates for tables
- Fix packaging bug when installed as git checkout
- Fix filter_class related bug in list views

### 1.4.15 1.1.1 - 2018-08-27

- Remove local style fix for invalid form inputs, fixed in upstream django-tapeforms

### 1.4.16 1.1.0 - 2018-08-17

- Add support for Django 2.1
- Use auth class based views for login an logout instead of function based views.

### 1.4.17 1.0.0 - 2018-05-25

- Many bugfixes and small improvements
- Add CollectionMixin
- Add `get_titlebuttons` helper to add additional buttons to Cruditor views
- Refactor templates to use UIKit instead of Bootstrap 3
- Introduce build process for Javascript and CSS files
- Add support for formsets, including Javascript for the user interface

### 1.4.18 0.1.4

- Update translations.

### 1.4.19 0.1.3

- Add missing floppyforms load tag.

### 1.4.20 0.1.2

- Add floppyforms form tag to inline formset template.

### 1.4.21 0.1.1

- Added some useful template blocks.

### 1.4.22 0.1.0

- Initial release without many docs but an example project.

CHAPTER

TWO

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## C

cruditor.collection, 9
cruditor.filters, 5
cruditor.forms, 4
cruditor.mixins, 2
cruditor.views, 6

# INDEX


## A

add_fields() (*cruditor.forms.CruditorFormsetMixin method*), 4
AnyChoiceFilter (*class in cruditor.filters*), 5

## B

base_fields (*cruditor.forms.ChangePasswordForm attribute*), 5
base_fields (*cruditor.forms.LoginForm attribute*), 5

## C

change_password_url (*cruditor.mixins.CruditorMixin attribute*), 2
ChangePasswordForm (*class in cruditor.forms*), 5
collection_detail_urlname (*cruditor.collection.CollectionViewMixin attribute*), 9
collection_include_detail_crumb() (*cruditor.collection.CollectionViewMixin method*), 10
collection_include_list_crumb() (*cruditor.collection.CollectionViewMixin method*), 10
collection_list_title (*cruditor.collection.CollectionViewMixin attribute*), 9
collection_list_urlname (*cruditor.collection.CollectionViewMixin attribute*), 9
CollectionViewMixin (*class in cruditor.collection*), 9
cruditor.collection
    module, 9
cruditor.filters
    module, 5
cruditor.forms
    module, 4
cruditor.mixins
    module, 2
cruditor.views
    module, 6
Cruditor403View (*class in cruditor.views*), 6
Cruditor404View (*class in cruditor.views*), 6
CruditorAddView (*class in cruditor.views*), 7
CruditorChangePasswordView (*class in cruditor.views*), 8
CruditorChangeView (*class in cruditor.views*), 8
CruditorDeleteView (*class in cruditor.views*), 8
CruditorFormsetFormMixin (*class in cruditor.forms*), 5
CruditorFormsetMixin (*class in cruditor.forms*), 4
CruditorListView (*class in cruditor.views*), 6
CruditorLogoutView (*class in cruditor.views*), 9
CruditorMixin (*class in cruditor.mixins*), 2
CruditorTapeformMixin (*class in cruditor.forms*), 4

## D

declared_fields (*cruditor.forms.ChangePasswordForm attribute*), 5
declared_fields (*cruditor.forms.LoginForm attribute*), 5
default_lookup_type (*cruditor.filters.MultiCharFilter attribute*), 6
delete() (*cruditor.views.CruditorDeleteView method*), 8
dispatch() (*cruditor.mixins.CruditorMixin method*), 2
dispatch() (*cruditor.views.Cruditor403View method*), 6
dispatch() (*cruditor.views.Cruditor404View method*), 6

## E

ensure_logged_in() (*cruditor.mixins.CruditorMixin method*), 3
ensure_logged_in() (*cruditor.views.CruditorLogoutView method*), 9
ensure_required_permission() (*cruditor.mixins.CruditorMixin method*), 3
extrahead_template_name (*cruditor.mixins.CruditorMixin attribute*), 2

## F

filter() (*cruditor.filters.MultiCharFilter method*), 6

filter_class (*cruditor.views.CruditorListView attribute*), 7
form_class (*cruditor.views.CruditorChangePasswordView attribute*), 9
form_invalid() (*cruditor.mixins.FormViewMixin method*), 4
form_valid() (*cruditor.mixins.FormViewMixin method*), 4
form_valid() (*cruditor.views.CruditorChangePasswordView method*), 9
form_valid() (*cruditor.views.CruditorDeleteView method*), 8
format_linked_objects() (*cruditor.views.CruditorDeleteView method*), 8
formset_classes (*cruditor.mixins.FormViewMixin attribute*), 3
FormViewMixin (*class in cruditor.mixins*), 3

## G

get() (*cruditor.mixins.FormViewMixin method*), 4
get_breadcrumb() (*cruditor.collection.CollectionViewMixin method*), 10
get_breadcrumb() (*cruditor.mixins.CruditorMixin method*), 3
get_breadcrumb_title() (*cruditor.collection.CollectionViewMixin method*), 9
get_breadcrumb_title() (*cruditor.mixins.CruditorMixin method*), 3
get_collection_detail_title() (*cruditor.collection.CollectionViewMixin method*), 10
get_collection_detail_url() (*cruditor.collection.CollectionViewMixin method*), 10
get_collection_list_title() (*cruditor.collection.CollectionViewMixin method*), 10
get_collection_list_url() (*cruditor.collection.CollectionViewMixin method*), 10
get_context_data() (*cruditor.mixins.CruditorMixin method*), 3
get_context_data() (*cruditor.views.CruditorChangePasswordView method*), 9
get_context_data() (*cruditor.views.CruditorChangeView method*), 8
get_context_data() (*cruditor.views.CruditorListView method*), 7
get_context_data() (*cruditor.views.CruditorLogoutView method*), 9
get_cruditor_context() (*cruditor.mixins.CruditorMixin method*), 2
get_delete_url() (*cruditor.views.CruditorChangeView method*), 8
get_filter_class() (*cruditor.views.CruditorListView method*), 7
get_filter_kwargs() (*cruditor.views.CruditorListView method*), 7
get_filtered_queryset() (*cruditor.views.CruditorListView method*), 7
get_form_kwargs() (*cruditor.views.CruditorChangePasswordView method*), 9
get_formset_classes() (*cruditor.mixins.FormViewMixin method*), 3
get_formset_kwargs() (*cruditor.mixins.FormViewMixin method*), 3
get_js_formset_options() (*cruditor.forms.CruditorFormsetMixin method*), 4
get_model_verbose_name() (*cruditor.mixins.CruditorMixin method*), 3
get_object() (*cruditor.views.CruditorAddView method*), 7
get_queryset() (*cruditor.views.CruditorListView method*), 7
get_required_permission() (*cruditor.mixins.CruditorMixin method*), 3
get_success_message() (*cruditor.mixins.FormViewMixin method*), 4
get_table() (*cruditor.views.CruditorListView method*), 7
get_table_class() (*cruditor.collection.CollectionViewMixin method*), 10
get_table_class() (*cruditor.views.CruditorListView method*), 7
get_table_kwargs() (*cruditor.views.CruditorListView method*), 7
get_template_context() (*cruditor.forms.CruditorFormsetMixin method*), 4
get_title() (*cruditor.collection.CollectionViewMixin method*), 9
get_title() (*cruditor.mixins.CruditorMixin method*), 2
get_title() (*cruditor.views.CruditorAddView method*), 8
get_title() (*cruditor.views.CruditorChangeView method*), 8
get_title() (*cruditor.views.CruditorDeleteView method*), 8
get_titlebuttons() (*cruditor.mixins.CruditorMixin*

*method*), 3

## H

handle_not_logged_in() (*cruditor.mixins.CruditorMixin method*), 3
hidden_fields() (*cruditor.forms.CruditorFormsetFormMixin method*), 5

## I

index_url (*cruditor.mixins.CruditorMixin attribute*), 2

## J

js_formset_options (*cruditor.forms.CruditorFormsetMixin attribute*), 4

## L

login_form_class (*cruditor.mixins.CruditorMixin attribute*), 2
login_template_name (*cruditor.mixins.CruditorMixin attribute*), 2
LoginForm (*class in cruditor.forms*), 5
logout_url (*cruditor.mixins.CruditorMixin attribute*), 2
lookup_types (*cruditor.filters.MultiCharFilter attribute*), 6

## M

media (*cruditor.forms.ChangePasswordForm property*), 5
media (*cruditor.forms.LoginForm property*), 5
menu_template_name (*cruditor.mixins.CruditorMixin attribute*), 2
menu_title (*cruditor.mixins.CruditorMixin attribute*), 2
model (*cruditor.views.CruditorListView attribute*), 6
model_verbose_name (*cruditor.mixins.CruditorMixin attribute*), 2
module
    cruditor.collection, 9
    cruditor.filters, 5
    cruditor.forms, 4
    cruditor.mixins, 2
    cruditor.views, 6
MultiCharFilter (*class in cruditor.filters*), 5

## P

perform_delete() (*cruditor.views.CruditorDeleteView method*), 8
post() (*cruditor.mixins.FormViewMixin method*), 4

## Q

queryset (*cruditor.views.CruditorListView attribute*), 7

## R

required_permission (*cruditor.mixins.CruditorMixin attribute*), 2

## S

save_form() (*cruditor.mixins.FormViewMixin method*), 4
staff_required (*cruditor.mixins.CruditorMixin attribute*), 2
success_message (*cruditor.views.CruditorAddView attribute*), 7
success_message (*cruditor.views.CruditorChangeView attribute*), 8
success_message (*cruditor.views.CruditorDeleteView attribute*), 8

## T

table_class (*cruditor.views.CruditorListView attribute*), 7
template_context (*cruditor.forms.CruditorFormsetMixin attribute*), 4
template_name (*cruditor.views.Cruditor403View attribute*), 6
template_name (*cruditor.views.Cruditor404View attribute*), 6
template_name (*cruditor.views.CruditorAddView attribute*), 7
template_name (*cruditor.views.CruditorChangePasswordView attribute*), 8
template_name (*cruditor.views.CruditorChangeView attribute*), 8
template_name (*cruditor.views.CruditorDeleteView attribute*), 8
template_name (*cruditor.views.CruditorListView attribute*), 7
template_name (*cruditor.views.CruditorLogoutView attribute*), 9
title (*cruditor.views.CruditorChangePasswordView attribute*), 9

## V

visible_fields() (*cruditor.forms.CruditorFormsetFormMixin method*), 5