CRUDLFA+ Documentation

Release 0.0

James Pic & Contributors

Oct 17, 2019

Contents:

1 Install CRUDLFA+ module **3**
- 1.1 Installing from PyPi . . . 3
- 1.2 With development packages . . . 3
- 1.3 Installing from GitHub . . . 3
- 1.4 Installing from source . . . 4

2 CRUDLFA+ Tutorial **5**
- 2.1 About document . . . 5
- 2.2 About module . . . 5
- 2.3 Enable in your project . . . 5
- 2.4 Define a Router . . . 6
- 2.5 URLs . . . 6
- 2.6 Changing home page . . . 7

3 Route class **9**

4 URLPatterns autogeneration mechanisms: Router **13**
- 4.1 Menus . . . 13

5 Settings **17**
- 5.1 Project . . . 17

6 Views **21**

7 Factory DRY patterns **23**

8 crudlfap_auth: crudlfap module for django.contrib.auth **25**
- 8.1 Auth Views . . . 25

9 Indices and tables **27**

Python Module Index **29**

Index **31**

CRUDLFA+ stands for Create Read Update Delete List Form Autocomplete and more.

This plugin for Django makes a rich user interface from Django models.

CHAPTER 1

Install CRUDLFA+ module

This section concerns This package can be installed from PyPi by running:

1.1 Installing from PyPi

If you are just getting started with CRUDLFA+, it is recommended that you start by installing the latest version from the Python Package Index (PyPi). To install CRUDLFA+ from PyPi using pip run the following command in your terminal.

```
pip install crudlfap
```

If you are not in a virtualenv, the above will fail if not executed as root, in this case use `install --user`:

```
pip install --user crudlfap
```

1.2 With development packages

If you intend to run the `crudlfap dev` command, then you should have the development dependencies by adding `[dev]`:

```
pip install (--user) crudlfap[dev]
```

Then, you should see the example project running on port 8000 with command:

```
crudlfap dev
```

1.3 Installing from GitHub

You can install the latest current trunk of crudlfap directly from GitHub using pip.

```
pip install --user -e git+git://github.com/yourlabs/crudlfap.git@master
↪#egg=crudlfap[dev]
```

Warning: `[dev]`, `--user`, `@master` are all optionnal above.

1.4 Installing from source

1. Download a copy of the code from GitHub. You may need to install git.

```
git clone https://github.com/yourlabs/crudlfap.git
```

2. Install the code you have just downloaded using pip, assuming your current working directory has not changed since the previous command it could be:

   ```
   pip install -e ./crudlfap[dev]
   ```

Move on to the *CRUDLFA+ Tutorial*.

CHAPTER 2

CRUDLFA+ Tutorial

2.1 About document

This document attempts to teach the patterns you can use, and at the same time go through every feature. The document strives to teach CRUDLFA+ as efficiently as possible. If it becomes too long, we will see how we refactor the document, until then, it serves as main documentation. Please contribute any modification you feel this document needs to fit its purpose.

2.2 About module

CRUDLFA+ strives to provide a modern UI for Django generic views out of the box, but all defaults should also be overiddable as conveniently as possible. It turns out that Django performs extremely well already, and pushing Django's philosophy such as DRY as far as possible works very well for me.

2.3 Enable in your project

We're going to setup `TEMPLATES` and `INSTALLED_APPS` before we begin.

Note: We will review the minimal settings in this tutorial, but you can consult the default settings available for your crudlfap version in the *`settings`* module.

2.3.1 TEMPLATES

CRUDLFA+ uses Jinja2 templates with a quite extended configuration. Options to enable them are using any of these in your settings:

- easiest: *`crudlfap.settings.TEMPLATES`*

- intermediate: *crudlfap.settings.CRUDLFAP_TEMPLATE_BACKEND*
- custom: *crudlfap.settings.DEFAULT_TEMPLATE_BACKEND*

2.3.2 INSTALLED_APPS

CRUDLFA+ leverages apps from the Django ecosystem. Use *crudlfap.settings.CRUDLFAP_TEMPLATE_BACKEND*. To help make this a pleasant experience, CRUDLFAP+ splits the INSTALLED_APPS setting into multiple settings you can import and mix together:

- everything: *crudlfap.settings.INSTALLED_APPS*,
- crudlfap only: *crudlfap.settings.CRUDLFAP_APPS*,
- django apps: *crudlfap.settings.DJANGO_APPS*,

2.4 Define a Router

2.4.1 Register a CRUD with default views using Router.register()

Just add a `crudlfap.py` file in one of your installed apps, and the `DefaultConfig` will autodiscover them, this example shows how to enable the default CRUD for a custom model:

```
from crudlfap import shortcuts as crudlfap

from .models import Artist


crudlfap.Router(
    Artist,
    fields='__all__',
).register()
```

In this case, the *Router* will get the views it should serve from the *CRUDLFAP_VIEWS* setting.

2.4.2 Custom view parameters with View.clone()

If you want to specify views in the router:

```
.. literalinclude:: ../src/crudlfap_example/song/crudlfap.py
```

Using the *clone()* classmethod will define a subclass on the fly with the given attributes.

2.5 URLs

The easiest configuration is to generate patterns from the default registry:

```
from crudlfap import shortcuts as crudlfap

urlpatterns = [
    crudlfap.site.urlpattern
]
```

Or, to sit in `/admin`:

```
crudlfap.site.urlpath = 'admin'

urlpatterns = [
    crudlfap.site.urlpattern,
    # your patterns ..
]
```

2.6 Changing home page

CRUDLFA+ so far relies on Jinja2 and provides a configuration where it finds templates in app_dir/jinja2.

As such, a way to override the home page template is to create a directory "jinja2" in one of your apps - personnaly i add the project itself to INSTALLED_APPS, sorry if you have hard feelings about it but i love to do that, have a place to put project-specific stuff in general - and in the *jinja2* directory create a *crudlfap/home.html* file.

You will also probably want to override *crudlfap/base.html*. But where it gets more interresting is when you replace the home view with your own. Example, still in urls.py:

```
from crudlfap import shortcuts as crudlfap
from .views import Dashboard  # your view

crudlfap.site.title = 'Your Title'  # used by base.html
crudlfap.site.urlpath = 'admin'  # example url prefix
crudlfap.site.views['home'] = views.Dashboard

urlpatterns = [
    crudlfap.site.get_urlpattern(),
]
```

So, there'd be other ways to acheive this but that's how i like to do it.

CHAPTER 3

Route class

CRUDLFA+ introduces an MVC-ish pattern, as the Router class is meant to sit between a Model class and its set of View. Your views will have to inherit from Route to work in Router.views. This structural decision made for you by CRUDLFA+ was not exactly designed: it's an open source rewrite of a module that was ordered in a proprietary project.

class `crudlfap.route.`**`Route`**

The mixin for Views that will make it compatible with Router.

`authenticate`

False by default, it makes the default has_perm() implementation require Django permission.

`urlargs`

Args that should be passed to reverse() along with Route.urlfullname.

`url`

Absolute url to the view, relying on Route.urlfullname and Route.urlargs.

You will be able to check if a user has access to a view with a given object for example as such:

```
crudlfap.site[YourModel]['detail'].clone(
    request=request,
    object=obj,
).has_perm()
```

If you want to open a View to all, set authenticate=False, examples:

```
class YourDetailView(DetailView):
    authenticate = False

class YourRouter(Router):
    views = [
        YourDetailView,
        ListView.clone(authenticate=False),  # example with clone
    ]
```

Without authenticate=False, the default has_perm() implementation requires the request user to have the permission corresponding to the permission_fullcode attribute.

To create the permission with permission_fullcode, you can browse in your CRUDLFA+ site and navigate to URL list view, for each URL you have link in the menu called "authorized" that lets you select which groups have this permission: it will auto-create the permission in the database if necessary.

dispatch(*request*, **args*, ***kwargs*)
This will run has_perm prior to super().dispatch().

get_permission_codename()
Return the codename attribute for the view Permission.

get_permission_fullcode()
Return a string with the app name, permission_shortcode and model name.

get_permission_shortcode()
Return the middle part for the view permission.

Returns the urlname by default.

get_url()
Return the URL for this view given its current state. Given that the `reverse()` method is a class method, this should allow things like:

```
url = YourView(object=your_object).url
```

get_urlargs()
Return args for reversing this view url from self. See `self.reverse()` for detail.

has_perm()
Checks for user permission.

classmethod reverse(**args*, ***kwargs*)
Reverse a url to this view with the given args.

class crudlfap.route.**RouteMetaclass**
Base autocalculations for views.

app_name
The view's app name.

model
The view's model if any.

urlpath
The path for the url path definition.

label
The view label, serves as key in a Router.views.

urlpattern
The Django URL path() instance, for inclusion in url lists.

urlfullname
The full name to reverse the URL, with namespaces if any.

urlfield
The default model field that will be use to match in the URL. It can be pk, or name, slug . . .

get_app_name()
Return the model's app_name or None.

get_label()
Return a readable label for this view.

Strips View and Route from class name, also removes the model class name if it finds it: for YourModelUpdateView this returns *update*.

get_model()
Return the router's model or None.

get_urlfield()
Return the router urlfield if any, else guess_urlfield()

get_urlfullname()
Return the url name eventually with router and site namespaces.

get_urlname()
Return a string that can be used as url name.

get_urlpath()
Return the urlname.

get_urlpattern()
Return the Django URL object to include in a urlpatterns.

CHAPTER 4

URLPatterns autogeneration mechanisms: Router

One of the key architectural concepts of CRUDLFA+ is the ability to tie a group of view with a model class to autogenerate urlpatterns. This chapter reviews the different mechanisms in place and how they are overridable.

Source is located in the `Router`, which we'll describe here.

The CRUDLFA+ Router is able to generate menus checking perms, generate urls . . .

Note: Note that you can also use non-database backed models, by inheriting from models.Model and setting their Meta.managed attribute to False. Then, you can use CRUDLFA+ views and routers.

4.1 Menus

A menu is referenced by a short name, and CRUDLFA+ generic views already define a bunch of them, but you can add your own too:

- `object`: means the view is for a model instance,
- `object_detail`: means the view should only be visible from detail view,
- `model`: means the view applies to a model class, such as list view,
- `main`: means the view should be in the main menu.

To get the views of a router, for a menu, kwargs such as the object, and with permissions on request.user use `Router.get_menu()`. In Jinja2 templates you can call them with:

```
{% set views=view.router.get_menu(
    'object',
    view.request,
    object=view.object
) %}
```

Now that Django can generate a menu after serious the refactoring that brought us to discover this pattern with Etienne Vidal @ DevNix, we rely on Jinja2 to refactor the HTML to render those menus.

The menu macro takes a list of views as argument, and also a unique HTML id it can use to generate the dropdown.

```
{% import 'crudlfap.html' as crudlfap %}
{{ crudlfap.dropdown(views, 'row-actions-' + str(object.pk)) }}
{# also works, different style: #}
{{ crudlfap.dropdownbutton(views, 'row-actions-' + str(object.pk)) }}
```

The above code will generate a Material design dropdown menu with an icon and the other one as a button with all nice icons, titles, permissions checked, and so on. This is used everywhere you see a part of the page that can spawn to a dropdown. If there is only one matching view, it will display only the button.

class `crudlfap.router.`**`Router`**(*model=None*, *registry=None*, *views=None*, ***attributes*)

Base router for CRUDLFA+ Route.

`model`

Optional model class for this Router and all its views.

`views`

`ViewsDescriptor` using *`CRUDLFAP_VIEWS`* by default, otherwise your list of views.

Note: The final views list is generated by the *`generate_views()`* method.

`generate_views`(**views*)

Generate views for this router, core of the automation in CRUDLFA+.

This method considers each view in given args or self.views and returns a list of usable views.

Each arg may be a view class or a dict of attributes with a *_cls* key for the actual view class.

It will copy the view class and bind the router on it in the list this returns.

For example, this would cause two view classes to be returned, if self.model is `Artist`, then `CreateView` will be used as parent to create `ArtistCreateView` and `DetailView` will be used to create `ArtistDetailView`, also setting the attribute `extra_stuff='bar'`:

```
Router(Artist).generate_views([
    CreateView,
    dict(_cls=DetailView, extra_stuff='bar'),
    ListView.factory(paginate_by=12),
])
```

`get_app_name`()

Generate app name for this Router views.

`get_fields`(*view*)

Return the list of fields for a user.

`get_menu`(*name*, *request*, ***kwargs*)

Return allowed view objects which have `name` in their `menus`.

For each view class in self.views which have `name` in their `menus` attribute, instanciate the view class with `request` and kwargs, call `has_perm()` on it.

Return the list of view instances for which `has_perm()` has passed.

`get_namespace`()

Generate namespace for this Router views.

get_queryset(*view*)
Return the queryset for a view, returns all by default.

get_urlfield()
Return Field name of model for reversing url.

This will return model ‘ slug ‘ field if available or ‘ pk ‘ field.

See `guess_urlfield()` for detail.

get_urlpath()
Return Model name for urlpath.

get_urlpatterns()
Generate URL patterns for this Router views.

has_perm(*view*)
View’s request.user has_perm call with the view’s permission_fullcode.

register()
Register to self.registry.

Also, adds the get_absolute_url() method to the model class if it has None, to return the reversed url for this instance to the view of this Router with the `detail` slug.

Set get_absolute_url in your model class to disable this feature. Until then, you got it for free.

Also, register this router as default router for its model class in the RouterRegistry.

class crudlfap.router.**Views**

CHAPTER 5

Settings

5.1 Project

A settings file to import boilerplate from.

`crudlfap.settings.`**`AUTHENTICATION_BACKENDS`**

Contains the default django.contrib.auth.backends.ModelBackend and also crudlfap_auth.backends.ViewBackend which will introspect the view's authenticate and allowed_groups variables.

`crudlfap.settings.`**`CRUDLFAP_VIEWS`**

List of default views to provide to Routers that were not spawned with any view.

`crudlfap.settings.`**`INSTALLED_APPS`**

That list contains both *`CRUDLFAP_APPS`* and *`DJANGO_APPS`* and you can use them as such on a new project:

```
from crudlfap.settings import INSTALLED_APPS

INSTALLED_APPS = ['yourapp'] + INSTALLED_APPS
```

`crudlfap.settings.`**`CRUDLFAP_APPS`**

List of apps CRUDLFA+ depends on, you can use it as such:

```
from crudlfap.settings import CRUDLFAP_APPS

INSTALLED_APPS = [
    'yourapp',
    'django.contrib.staticfiles',
    # etc
] + CRUDLFAP_APPS
```

`crudlfap.settings.`**`DJANGO_APPS`**

This list contains all contrib apps from the Django project that CRUDLFA+ should depend on. You can use it as such:

```
from crudlfap.settings import CRUDLFAP_APPS, DJANGO_APPS

INSTALLED_APPS = ['yourapp'] + CRUDLFAP_APPS + DJANGO_APPS
```

`crudlfap.settings.`**`TEMPLATES`**

This list contains both *DEFAULT_TEMPLATE_BACKEND* and *CRUDLFAP_TEMPLATE_BACKEND* and works out of the box on an empty project. You can add it to your settings file by just importing it:

```
from crudlfap.settings import TEMPLATES
```

`crudlfap.settings.`**`CRUDLFAP_TEMPLATE_BACKEND`**

Configuration for Jinja2 and environment expected by CRUDLFA+ default templates. Add it to your own TEMPLATES setting using import:

```
from crudlfap.settings import CRUDLFAP_TEMPLATE_BACKEND

TEMPLATES = [
    # YOUR_BACKEND
    CRUDLFAP_TEMPLATE_BACKEND,
]
```

`crudlfap.settings.`**`DEFAULT_TEMPLATE_BACKEND`**

Configuration for Django template backend with all builtin context processors. You can use it to define only your third backend as such:

```
from crudlfap.settings import (
    CRUDLFAP_TEMPLATE_BACKEND,
    DEFAULT_TEMPLATE_BACKEND,
)

TEMPLATES = [
   # YOUR_BACKEND
   CRUDLFAP_TEMPLATE_BACKEND,
   DEFAULT_TEMPLATE_BACKEND,
]
```

`crudlfap.settings.`**`DEBUG`**

Evaluate `DEBUG` env var as boolean, False by default.

`crudlfap.settings.`**`SECRET_KEY`**

Get `SECRET_KEY` env var, or be `'notsecret'` by default.

Danger: Raises an Exception if it finds both SECRET_KEY=notsecret and DEBUG=False.

`crudlfap.settings.`**`ALLOWED_HOSTS`**

Split `ALLOWED_HOSTS` env var with commas, or be `['*']` by default.

Danger: Raises an Exception if it finds both ALLOWED_HOSTS to be `'*'` and DEBUG=False.

`crudlfap.settings.`**`MIDDLEWARE`**

A default MIDDLEWARE configuration you can import.

`crudlfap.settings.`**`OPTIONAL_APPS`**

from crudlfap.settings import * # [. . .] your settings install_optional(OPTIONAL_APPS, INSTALLED_APPS) install_optional(OPTIONAL_MIDDLEWARE, MIDDLEWARE)

CHAPTER 6

Views

Source is located in the `generic`, which we'll describe here.

Crudlfa+ generic views and mixins.

Crudlfa+ takes views further than Django and are expected to:

- generate their URL definitions and reversions,
- check if a user has permission for an object,
- declare the names of the navigation menus they belong to.

class `crudlfap.views.generic.`**`CreateView`**`(`***kwargs*`)`
    View to create a model object.

class `crudlfap.views.generic.`**`DeleteObjectsView`**`(`***kwargs*`)`
    Delete selected objects.

class `crudlfap.views.generic.`**`DeleteView`**`(`***kwargs*`)`
    View to delete an object.

class `crudlfap.views.generic.`**`DetailView`**`(`***kwargs*`)`
    Templated model object detail view which takes a field option.

class `crudlfap.views.generic.`**`FormView`**`(`***kwargs*`)`
    Base FormView class.

class `crudlfap.views.generic.`**`HistoryView`**`(`***kwargs*`)`

class `crudlfap.views.generic.`**`ListView`**`(`***kwargs*`)`

class `crudlfap.views.generic.`**`ModelFormView`**`(`***kwargs*`)`

class `crudlfap.views.generic.`**`ModelView`**`(`***kwargs*`)`

class `crudlfap.views.generic.`**`ObjectFormView`**`(`***kwargs*`)`
    Custom form view on an object.

class `crudlfap.views.generic.`**`ObjectView`**`(`***kwargs*`)`

class `crudlfap.views.generic.`**`ObjectsFormView`**`(`***kwargs*`)`

class `crudlfap.views.generic.`**`ObjectsView`**(***kwargs*)

class `crudlfap.views.generic.`**`TemplateView`**(***kwargs*)
TemplateView for CRUDLFA+.

class `crudlfap.views.generic.`**`UpdateView`**(***kwargs*)
Model update view.

class `crudlfap.views.generic.`**`View`**(***kwargs*)
Base view for CRUDLFA+.

CHAPTER 7

Factory DRY patterns

CRIMINALLY INVASIVE HACKS in `Factory`.

class `crudlfap.factory.`**`Factory`**

Adds clumsy but automatic getter resolving.

The *__getattr__* override makes this class try to call a get_*() method for variables that are not in *self.__dict__*.

For example, when *self.foo* is evaluated *and 'foo' not in self.__dict__* then it will call the *self.get_foo()*

If *self.get_foo()* returns None, it will try to get the result again from *self.__dict__*. Which means that we are going to witness this horrroorr:

```
class YourEvil(Factory):
    def get_foo(self):
        self.calls += 1
        self.foo = 13

assert YourEvil.foo == 13    # crime scene 1
assert YourEvil.foo == 13    # crime scene 2
assert YourEvil.calls == 1   # crime scene 3
```

For the moment it is pretty clumsy because i tried to contain the criminality rate as low as possible meanwhile i like the work it does for me !

classmethod clone(**mixins*, ***attributes*)

Return a subclass with the given attributes.

If a model is found, it will prefix the class name with the model.

class `crudlfap.factory.`**`FactoryMetaclass`**

`__getattr__` that ensures a first argument to getters.

Makes the getter work both from class and instance

Thanks to this, your *get_*()* methods will /maybe/ work in both cases:

```
YourClass.foo   # calls get_foo(YourClass)
YourClass().foo # calls get_foo(self)
```

Don't code drunk.

get_cls()

Return the cls.

did it go to far at this point ?

CHAPTER 8

crudlfap_auth: crudlfap module for django.contrib.auth

8.1 Auth Views

Source is located in the `views`, which we'll describe here.

Crudlfa+ PasswordView, Become and BecomeUser views.

Crudlfa+ takes views further than Django and are expected to:

- generate their URL definitions and reversions,
- check if a user has permission for an object,
- declare the names of the navigation menus they belong to.

class `crudlfap_auth.views.`**`Become`**`(`***kwargs*`)`

> **`has_perm`**`()`
>
> Checks for user permission.

class `crudlfap_auth.views.`**`BecomeUser`**`(`***kwargs*`)`

> **`get_object`**`(`*queryset=None*`)`
>
> Return the object the view is displaying.
>
> Require *self.queryset* and a *pk* or *slug* argument in the URLconf. Subclasses can override this to return any object.
>
> **`get_title_menu`**`()`
>
> Return title for menu links to this view.

class `crudlfap_auth.views.`**`PasswordView`**`(`***kwargs*`)`

> **`get_form_kwargs`**`()`
>
> Return the keyword arguments for instantiating the form.

get_title_submit()

Title of the submit button.

Defaults to `title_menu`

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

c

crudlfap.factory, 23
crudlfap.route, 9
crudlfap.router, 13
crudlfap.settings, 17
crudlfap.views.generic, 21
crudlfap_auth.views, 25

Index

A

ALLOWED_HOSTS (*in module crudlfap.settings*), 18
AUTHENTICATION_BACKENDS (*in module crudlfap.settings*), 17

B

Become (*class in crudlfap_auth.views*), 25
BecomeUser (*class in crudlfap_auth.views*), 25

C

clone() (*crudlfap.factory.Factory class method*), 23
CreateView (*class in crudlfap.views.generic*), 21
crudlfap.factory (*module*), 23
crudlfap.route (*module*), 9
crudlfap.router (*module*), 13
crudlfap.settings (*module*), 17
crudlfap.views.generic (*module*), 21
CRUDLFAP_APPS (*in module crudlfap.settings*), 17
crudlfap_auth.views (*module*), 25
CRUDLFAP_TEMPLATE_BACKEND (*in module crudlfap.settings*), 18
CRUDLFAP_VIEWS (*in module crudlfap.settings*), 17

D

DEBUG (*in module crudlfap.settings*), 18
DEFAULT_TEMPLATE_BACKEND (*in module crudlfap.settings*), 18
DeleteObjectsView (*class in crudlfap.views.generic*), 21
DeleteView (*class in crudlfap.views.generic*), 21
DetailView (*class in crudlfap.views.generic*), 21
dispatch() (*crudlfap.route.Route method*), 10
DJANGO_APPS (*in module crudlfap.settings*), 17

F

Factory (*class in crudlfap.factory*), 23
FactoryMetaclass (*class in crudlfap.factory*), 23
FormView (*class in crudlfap.views.generic*), 21

G

generate_views() (*crudlfap.router.Router method*), 14
get_app_name() (*crudlfap.route.RouteMetaclass method*), 10
get_app_name() (*crudlfap.router.Router method*), 14
get_cls() (*crudlfap.factory.FactoryMetaclass method*), 24
get_fields() (*crudlfap.router.Router method*), 14
get_form_kwargs() (*crudlfap_auth.views.PasswordView method*), 25
get_label() (*crudlfap.route.RouteMetaclass method*), 10
get_menu() (*crudlfap.router.Router method*), 14
get_model() (*crudlfap.route.RouteMetaclass method*), 11
get_namespace() (*crudlfap.router.Router method*), 14
get_object() (*crudlfap_auth.views.BecomeUser method*), 25
get_permission_codename() (*crudlfap.route.Route method*), 10
get_permission_fullcode() (*crudlfap.route.Route method*), 10
get_permission_shortcode() (*crudlfap.route.Route method*), 10
get_queryset() (*crudlfap.router.Router method*), 14
get_title_menu() (*crudlfap_auth.views.BecomeUser method*), 25
get_title_submit() (*crudlfap_auth.views.PasswordView method*), 25
get_url() (*crudlfap.route.Route method*), 10
get_urlargs() (*crudlfap.route.Route method*), 10
get_urlfield() (*crudlfap.route.RouteMetaclass method*), 11
get_urlfield() (*crudlfap.router.Router method*), 15
get_urlfullname() (*crudlfap.route.RouteMetaclass method*), 11
get_urlname() (*crudlfap.route.RouteMetaclass*

method), 11
get_urlpath() (*crudlfap.route.RouteMetaclass method*), 11
get_urlpath() (*crudlfap.router.Router method*), 15
get_urlpattern() (*crudlfap.route.RouteMetaclass method*), 11
get_urlpatterns() (*crudlfap.router.Router method*), 15

H

has_perm() (*crudlfap.route.Route method*), 10
has_perm() (*crudlfap.router.Router method*), 15
has_perm() (*crudlfap_auth.views.Become method*), 25
HistoryView (*class in crudlfap.views.generic*), 21

I

INSTALLED_APPS (*in module crudlfap.settings*), 17

L

ListView (*class in crudlfap.views.generic*), 21

M

MIDDLEWARE (*in module crudlfap.settings*), 18
model (*crudlfap.router.Router attribute*), 14
ModelFormView (*class in crudlfap.views.generic*), 21
ModelView (*class in crudlfap.views.generic*), 21

O

ObjectFormView (*class in crudlfap.views.generic*), 21
ObjectsFormView (*class in crudlfap.views.generic*), 21
ObjectsView (*class in crudlfap.views.generic*), 22
ObjectView (*class in crudlfap.views.generic*), 21
OPTIONAL_APPS (*in module crudlfap.settings*), 18

P

PasswordView (*class in crudlfap_auth.views*), 25

R

register() (*crudlfap.router.Router method*), 15
reverse() (*crudlfap.route.Route class method*), 10
Route (*class in crudlfap.route*), 9
Route.authenticate (*in module crudlfap.route*), 9
Route.url (*in module crudlfap.route*), 9
Route.urlargs (*in module crudlfap.route*), 9
RouteMetaclass (*class in crudlfap.route*), 10
RouteMetaclass.app_name (*in module crudlfap.route*), 10
RouteMetaclass.label (*in module crudlfap.route*), 10
RouteMetaclass.model (*in module crudlfap.route*), 10
RouteMetaclass.urlfield (*in module crudlfap.route*), 10
RouteMetaclass.urlfullname (*in module crudlfap.route*), 10
RouteMetaclass.urlpath (*in module crudlfap.route*), 10
RouteMetaclass.urlpattern (*in module crudlfap.route*), 10
Router (*class in crudlfap.router*), 14

S

SECRET_KEY (*in module crudlfap.settings*), 18

T

TEMPLATES (*in module crudlfap.settings*), 18
TemplateView (*class in crudlfap.views.generic*), 22

U

UpdateView (*class in crudlfap.views.generic*), 22

V

View (*class in crudlfap.views.generic*), 22
Views (*class in crudlfap.router*), 15
views (*crudlfap.router.Router attribute*), 14