Django Hyperadmin Documentation

Release 0.10c0

Jason Kraus

October 25, 2013

CONTENTS

django-hyperadmin is a HATEOAS API framework for building resources in Django. Resources are anything that can be manipulated with forms and configuration of a Resource is similar to that of an Admin Model. While Resources offer a REST CRUD layer out of the box they are meant to power workflows that go beyond CRUD. Additionally Resources aim to be reusable throughout the web application and not to be limitted to a single API endpoint.

Download: http://github.com/webcube/django-hyperadmin

CHAPTER

ONE

TABLE OF CONTENTS

1.1 Installation

1.1.1 Requirements

- Python 2.6 or later
- Django 1.3 or later
- django-datatap

1.1.2 Settings

Put 'hyperadmin' into your `INSTALLED_APPS` section of your settings file.

And the following to urls.py:

```
import hyperadmin
hyperadmin.autodiscover() #TODO this does nothing
hyperadmin.site.install_models_from_site(admin.site) #ports admin models to hyperadmin
hyperadmin.site.install_storage_resources() #enables the storage resource for media and static
```

Add to root url patterns:

```
url(r'^hyperapi/', include(hyperadmin.site.urls)),
```

(Optional) Install a client:

- *Clients*

(Optional) Install additional authenticators:

- *Authentication*

1.2 Introduction

Hyperadmin attempts to be an API framework capable of expressing HATEOAS accross various hypermedia formats.

1.2.1 Links

A link is an available state transition that can be performed over HTTP. Links may also have a form attached to represent the various parameters the user may provide. Links are provided by endpoints and drive Hypermeda as the Engine of Application State. A link may require an HTTP method that is not supported by HTML and it is recommended that an endpoint provide an alternative link to support pure HTML clients.

1.2.2 Serialization

Serialization is done from a form object that is attached to an *Item*. Endpoints translate items from their data store to an appropriate form so the media types do not need to know anything about the data store. This way arbitrary storage backends may be added without modifying the media type implementations. Rather media types concentrate on serializing forms and links.

1.2.3 Endpoints

Endpoints build state and provide links at a given URL. Endpoints may contain child endpoints or implement a class based view to handle an incomming request.

1.2.4 Resources

Resources are a collection of endpoints that describe a particular service.

1.2.5 State

All API requests have a state generated and associated to them. A state contains the items to be represented in the response, filtering params, and other general session and request variables to be shared accross the request cycle.

1.2.6 Root Endpoint

A root endpoint is the top most level of an API. It does not provide links but manages authentication and media type registration.

1.3 Authentication

1.3.1 API Key

hyperadmin ships with an application to do basic API key authentication.

Installation

Add *hyperadmin.contrib.apikey* to INSTALLED_APPS.

Add the following to your settings:

```
DEFAULT_API_REQUEST_CLASS = 'hyperadmin.contrib.apikey.apirequests.HTTPAPIKeyRequest'
```

Or to explicitly set the api request class of a site:

```
from hyperadmin.contrib.apikey.apirequests import HTTPAPIKeyRequest
site = ResourceSite(apirequest_class=HTTPAPIKeyRequest)
```

1.4 Clients

Visiting the api endpoint in a browser will let you browse the various hyberobjects made available through the resource. Clients may be installed on a different url.

1.4.1 Django Template Client

https://github.com/webcube/django-hyperadmin-client

Uses django templates to render an admin interface. Responsive design out of the box.

1.4.2 Ember REST Client

https://github.com/zbyte64/django-hyperadmin-emberclient

Uses REST calls and emberjs to render an admin interface.

1.4.3 Backbone Bindings

https://github.com/zbyte64/django-hyperadmin-backboneclient

Provides basic bindings to the Backbone API.

1.4.4 Dockit CMS

https://github.com/webcube/django-dockitcms

A dynamic API builder with a public HTML (template driven) client.

1.5 Wizards

The Wizard resource is a collection of step endpoints which represent a series of ordered api requests needed to complete a single task. Once a wizard is created in can be included in your urls or registered to a resource site.

Example:

```
from django import forms

from hyperadmin.resources.wizard import Wizard, FormStep


class EmailForm(forms.Form):
    email = forms.EmailField()

class UsernameForm(forms.Form):
    username = forms.CharField()
```

```
class PasswordForm(forms.Form):
    password = forms.CharField()

class GetEmail(FormStep):
    form_class = EmailForm

class GetUsername(FormStep):
    form_class = UsernameForm

    def can_skip(self):
        return True

class GetPassword(FormStep):
    form_class = PasswordForm

class GetAttribute(FormStep):
    form_class = AttributeForm

    def can_skip(self):
        return True

class SimpleWizard(Wizard):
    step_definitions = [
        (GetEmail, {'slug':'email'}),
        (GetUsername, {'slug':'username'}),
        (GetPassword, {'slug':'password'}),
    ]

    def done(self, submissions):
        return 'success'

wizard = SimpleWizard(app_name='users', resource_adaptor='make_user')
urlpatterns += patterns('', (r'^', include(wizard.urls)))
```

A multi-part step allows for multiple steps to be embedded in a single step.

Multi part step example:

```
from django import forms

from hyperadmin.resources.wizard import Wizard, FormStep, MultiPartStep


class EmailForm(forms.Form):
    email = forms.EmailField()

class UsernameForm(forms.Form):
    username = forms.CharField()

class PasswordForm(forms.Form):
    password = forms.CharField()

class AttributeForm(forms.Form):
    key = forms.CharField()
    value = forms.CharField()

class GetEmail(FormStep):
    form_class = EmailForm
```

```
class GetUsername(FormStep):
    form_class = UsernameForm

class GetPassword(FormStep):
    form_class = PasswordForm

class GetAttribute(FormStep):
    form_class = AttributeForm

    def can_skip(self):
        return True

class GetAttributes(MultiPartStep):
    step_definitions = [
        (GetAttribute, {'slug':'attr1'}),
        (GetAttribute, {'slug':'attr2'}),
    ]

class ExpandedWizard(Wizard):
    step_definitions = [
        (GetEmail, {'slug':'email'}),
        (GetUsername, {'slug':'username'}),
        (GetPassword, {'slug':'password'}),
        (GetAttributes, {'slug':'attributes'}),
    ]

    def done(self, submissions):
        return 'success'
```

1.6 Content Types

Hyperadmin supports 3 different modes of content types:

- HTML - renders a response using the django template engine
- Hypermedia - renders a structured response representing a workflow and data (ie application/vnd.collection+json)
- Datatap - renders and loads fixtures (ie application/json)

1.6.1 Media Type Selection

The incomming media type is handled seperately from the outgoing media type. The selection of each media type can be controlled with the following HTTP headers:

Accept: Specifies the media type of the response

Content-Type: Specifies the media type of the request

Alternatively they may also be controlled by the following GET parameters:

_HTTP_ACCEPT: Specifies the media type of the response

_CONTENT_TYPE: Specifies the media type of the request

Note: Arguments passed by the GET method needs to be URL encoded. In JavaScript you can use the built-in *encodeURIComponent* method like so:

```
var contentType = encodeURIComponent('application/vnd.Collection.hyperadmin+JSON')
```

1.7 3rd Party Resources

1.7.1 S3 Storage

Storage resource that enables direct uploading to S3:

https://github.com/webcube/django-hyperadmin-s3resource

1.7.2 Dockit

CRUD resource for integrating with django-dockit:

https://github.com/webcube/django-hyperadmin-dockitresource

1.8 Internal API Reference

1.8.1 Resources

Resources

BaseResource

Methods

class `hyperadmin.resources.resources.`**`BaseResource`**(***kwargs*)

Bases: `hyperadmin.endpoints.GlobalSiteMixin`, `hyperadmin.endpoints.VirtualEndpoint`

A collection of endpoints representing a particular service

`resource_class` = ''

`form_class`

alias of `EmptyForm`

`resource_item_class`

alias of `ResourceItem`

`name_suffix` = 'resource'

`resource_adaptor` = None

The object representing the resource connection. Typically passed in during construction

`fork`(***kwargs*)

`resource`

`post_register`()

`get_app_name`()

Return the application name of this resource. Provides the return value of the *app_name* property.

`app_name`

Set or get the application name

get_resource_name()
Return the name of this resource. Provides the return value of the *resource_name* property.

resource_name
Set or get the name of the resource

get_resource_slug()
Return the slug of this resource. Provides the return value of the *resource_slug* property.

resource_slug
Set or get the slug of the resource

get_prompt()

get_base_url_name_suffix()

register_endpoints()

register_endpoint(*endpoint_cls*, ***kwargs*)

get_view_endpoints()
Returns a list of tuples containing (endpoint class, endpoint kwargs)

get_children_endpoints()

reverse(*name*, **args*, ***kwargs*)

get_state_data()

get_indexes()

get_index(*name*)

get_index_query(*name*)

get_item_url(*item*)

get_related_resource_from_field(*field*)

get_html_type_from_field(*field*)

get_absolute_url()

get_url(***kwargs*)

get_resource_link_item()

get_index_endpoint()

get_breadcrumb()

get_breadcrumbs()

get_paginator_kwargs()

emit_event(*event*, *item_list=None*)
Fires of the *resource_event* signal

ResourceEndpoint

Methods

class hyperadmin.resources.endpoints.**ResourceEndpoint**(***kwargs*)
Bases: hyperadmin.endpoints.Endpoint

resource

create_link_collection()

get_resource_item(*instance*, ***kwargs*)

get_instances()

get_common_state()

common_state

get_resource_link(***kwargs*)

get_item_url(*item*)

get_item_prompt(*item*)

get_form_class()

get_form_kwargs(***kwargs*)

get_item_form_class()

get_item_form_kwargs(***kwargs*)

get_namespaces()

get_item_namespaces(*item*)

get_item_link(*item*)

get_breadcrumbs()

api_permission_check(*api_request*, *endpoint*)

create_apirequest(***kwargs*)

expand_template_names(*suffixes*)

get_context_data(***kwargs*)

get_datatap(***kwargs*)

ResourceItem

Methods

class hyperadmin.resources.hyperobjects.**ResourceItem**(*endpoint*, *instance*, *datatap=None*)

Bases: hyperadmin.hyperobjects.Item

resource

Model Resources

ModelResource

Registering models Registering a model with hyperadmin:

```
import hyperadmin
from hpyeradmin.resources.models import ModelResource, InlineModelResource
from myapp.models import MyModel, ChildModel

class ChildModelResource(InlineModelResource):
    model = ChildModel
```

```
class MyModelResource(ModelResource):
    inlines = [ChildModelResource]
    list_display = ['name', 'number']
    list_filter = ['timestamp', 'category']

hyperadmin.site.register(MyModel, MyModelResource)
```

Options

- model
- fields
- exclude
- paginator
- list_display
- list_filter (basic filters, don't know about custom)
- search_fields
- list_per_page
- form_class
- inlines

The params queryset, ordering, search_fields, list_select_related and date_hierarchy are planned.

Autoloaded Options These options are mapped to the ModelResource when using autoload.

- model
- fields
- fieldsets (flattened to provide fields)
- exclude
- paginator
- list_display
- list_filter (basic filters, don't know about custom)
- search_fields
- list_per_page
- form_class
- inlines

API Endpoints

- "/" lists rows; POST to create
- "/add/" POST to add
- "/<id>/" displays a specific row; PUT/POST to update, DELETE to delete
- "/<id>/delete/" POST to delete

Methods

class hyperadmin.resources.models.**ModelResource**(***kwargs*)

Bases: hyperadmin.resources.models.resources.BaseModelResource

list_endpoint = (<class 'hyperadmin.resources.crud.endpoints.ListEndpoint'>, {'index_name': 'filter'})

create_endpoint = (<class 'hyperadmin.resources.crud.endpoints.CreateEndpoint'>, {})

detail_endpoint = (<class 'hyperadmin.resources.crud.endpoints.DetailEndpoint'>, {})

delete_endpoint = (<class 'hyperadmin.resources.crud.endpoints.DeleteEndpoint'>, {})

post_register()

model

initialize_inlines()

register_inline(*inline_cls*)

get_urls()

get_item_namespaces(*item*)

InlineModelResource

Methods

class hyperadmin.resources.models.**InlineModelResource**(*parent*, ***kwargs*)

Bases: hyperadmin.resources.models.resources.BaseModelResource

model = None

fk_name = None

rel_name = None

list_endpoint = (<class 'hyperadmin.resources.models.endpoints.InlineListEndpoint'>, {})

create_endpoint = (<class 'hyperadmin.resources.models.endpoints.InlineCreateEndpoint'>, {})

detail_endpoint = (<class 'hyperadmin.resources.models.endpoints.InlineDetailEndpoint'>, {})

delete_endpoint = (<class 'hyperadmin.resources.models.endpoints.InlineDeleteEndpoint'>, {})

post_register()

get_queryset(*parent*)

get_primary_query(***kwargs*)

get_indexes()

get_base_url_name_suffix()

get_item_url(*item*)

get_absolute_url()

get_breadcrumbs()

get_form_class()

get_ln_links()

get_idempotent_links()

API Endpoints

- "/<parent_id>/(relname)/" lists rows; POST to create
- "/<parent_id>/(relname)/add/" POST to add
- "/<parent_id>/(relname)/<id>/" displays a specific row; PUT/POST to update, DELETE to delete
- "/<parent_id>/(relname)/<id>/delete/" POST to delete

GenericInlineModelResource

Methods

class hyperadmin.resources.models.generic.**GenericInlineModelResource**(*parent*, ***kwargs*)

Bases: hyperadmin.resources.models.resources.InlineModelResource

model = None

ct_field = 'content_type'

ct_fk_field = 'object_id'

post_register()

content_type

get_queryset(*parent*)

get_primary_query(***kwargs*)

get_form_class()

Autoload Model Resources

Maps django model admin entries to model resources.

Usage:

```
from hyperadmin.resources.models.autload import DjangoCTModelAdminLoader
from django.contrib.admin import site as admin_site
from hyperadmin import site as root_endpoint

loader = DjangoCTModelAdminLoader(root_endpoint, admin_site)
loader.register_resources()
```

DjangoModelAdminLoader

class hyperadmin.resources.models.autload.**DjangoModelAdminLoader**(*root_endpoint*, *admin_site*)

A helper class that maps admin entries from a *django.contrib.admin.site.AdminSite* object to a *RootEndpoint*

get_logger()

register_resources()

generate_resource(*admin_model*)

When supplied a subclass of *ModelAdmin* Returns a ModelResource class with the following options mapped:

•fields

- fieldsets (flattened to provided fields)
- exclude
- paginator
- list_display
- list_filter
- list_select_related (not used)
- list_per_page
- list_max_show_all (not used)
- list_editable (not used)
- search_fields
- date_hierarchy (not used)
- ordering (not used)
- form_class

register_inlines(*admin_model*, *resource*)

register_inline(*admin_model*, *resource*, *inline_cls*)

generate_inline(*inline_cls*)
When supplied a subclass of *InlineModelAdmin* Returns an InlineModelResource with the following options mapped:

- model
- fields
- exclude
- fk_name

DjangoCTModelAdminLoader

class hyperadmin.resources.models.autoload.**DjangoCTModelAdminLoader**(*root_endpoint*, *admin_site*)
Bases: hyperadmin.resources.models.autoload.DjangoModelAdminLoader

Extends *DjangoModelAdminLoader* to provide support for autloading generic inlines

generate_inline(*inline_cls*)
When supplied a subclass of *GenericInlineModelAdmin* Returns a GenericInlineModelResource with the following options mapped:

- model
- fields
- exclude
- ct_field
- ct_fk_field

Directory Resources

ResourceDirectory

API Endpoints

- "/<key>/" where key belongs to the resource_adaptor

Methods

class hyperadmin.resources.directory.**ResourceDirectory**(***kwargs*)

Bases: hyperadmin.resources.resources.BaseResource

resource_class = 'resourcelisting'

form_class

alias of ViewResourceForm

list_endpoint_class

alias of ListEndpoint

get_view_endpoints()

get_urls()

register_resource(*resource*, *key=None*)

fork(***kwargs*)

get_instances()

get_item_prompt(*item*)

get_item_url(*item*)

get_item_outbound_links(*item*)

Auth Resource

Methods

class hyperadmin.resources.auth.**AuthResource**(***kwargs*)

Bases: hyperadmin.resources.resources.BaseResource

form_class

alias of AuthenticationResourceForm

login_endpoint_class

alias of LoginEndpoint

logout_endpoint_class

alias of LogoutEndpoint

get_view_endpoints()

api_permission_check(*api_request*, *endpoint*)

get_main_link_name()

get_index_endpoint()

StorageResource

Allows for direct browsing and uploading of files. The storage resource includes an endpoint for generating a direct upload link.

API Endpoints

- "/storages/media/" lists directories and files
- "/storages/media/?path=<path>" lists directories and files belonging to a certain path
- "/storages/media/upload-link/" POST to create a direct upload link
- "/storages/media/<path>/" endpoint for updating a particular file
- "/storages/static/"
- "/storages/static/<path>/"

Methods

class hyperadmin.resources.storages.**StorageResource**(***kwargs*)
Bases: hyperadmin.resources.crud.resources.CRUDResource

form_class
alias of UploadForm

upload_link_form_class
alias of UploadLinkForm

base64_upload_form_class
alias of Base64UploadForm

list_endpoint = (<class 'hyperadmin.resources.storages.endpoints.ListEndpoint'>, {})

create_upload_endpoint = (<class 'hyperadmin.resources.storages.endpoints.CreateUploadEndpoint'>, {})

base64_upload_endpoint = (<class 'hyperadmin.resources.storages.endpoints.Base64UploadEndpoint'>, {})

get_storage()

storage

get_base64_upload_form_class()

get_upload_link_form_class()

get_view_endpoints()

get_indexes()

get_primary_query()

get_instances()
Returns a set of native objects for a given state

get_item_form_kwargs(*item=None*, ***kwargs*)

get_form_kwargs(***kwargs*)

get_upload_link_form_kwargs(***kwargs*)

get_item_url(*item*)

get_item_storage_link(*item*, ***kwargs*)

get_item_outbound_links(*item*)

get_item_prompt(*item*)

get_paginator_kwargs()

Wizard Resources

Wizard

API Endpoints

- "/" loads the wizard, lists the substeps. POST to skip steps

Methods

class hyperadmin.resources.wizard.**Wizard**(***kwargs*)

Bases: hyperadmin.resources.resources.BaseResource

step_definitions = []

list_endpoint
alias of StepList

instance_form_class
alias of StepStatusForm

storage_class
alias of SessionStorage

get_index_endpoint()

storage

create_storage()

get_storage_class()

get_storage_kwargs()

steps

available_steps

step_index(*slug*)

get_instances()

get_item_url(*item*)

get_view_endpoints()

set_step_status(*slug*, *status*)

update_statuses(*statuses*)

step_statuses

get_step_statuses()

set_step_statuses(*statuses*)

get_step_data(*key*)

set_step_data(*key*, *value*)

get_next_step(*skip_steps*=[ ], *desired_step*=*None*)

next_step(*skip_steps*=[ ], *desired_step*=*None*)

done(*submissions*)

get_current_step_links()

get_context_data(***kwargs*)

FormStep

API Endpoints

- "/<slug>/" POST to submit step

Methods

class hyperadmin.resources.wizard.**FormStep**(***kwargs*)

Bases: hyperadmin.resources.wizard.endpoints.Step

link_prototype

alias of FormStepLinkPrototype

form_class = None

prototype_method_map

get_link_prototypes()

get_context_data(***kwargs*)

form_valid(*form*)

form_invalid(*form*)

MultiPartStep

API Endpoints

- "/<slug>/" POST to skip substeps
- "/<slug>/<substep>/" POST to submit substep

Methods

class hyperadmin.resources.wizard.**MultiPartStep**(***kwargs*)

Bases: hyperadmin.resources.wizard.endpoints.StepProvider, hyperadmin.resources.wizard.resources.Wizard

get_resource_name()

get_url_suffix()

get_base_url_name_suffix()

can_skip()

get_outbound_links()

done(*submissions*)

get_current_step_links()

expand_template_names(*suffixes*)

create_apirequest(***kwargs*)

1.8.2 Endpoints

BaseEndpoint

class hyperadmin.endpoints.**BaseEndpoint**(***kwargs*)

Represents an API Endpoint

form_class = None
The default form class for links created by this endpoint

item_form_class = None
The form class representing items from this endpoint. The form created is used for serialization of item.

state_class
alias of `EndpointState`

resource_item_class
alias of `Item`

app_name = None
The slug that identifies the application of this endpoint. Typically set by the site object.

base_url_name_suffix = None
The suffix to apply to the base url name when concating the base name from the parent

base_url_name = None
Set the base url name instead of getting it from the parent

name_suffix = None
The suffix to add to the generated url name. This also used by the parent endpoint to reference child endpoints

url_name = None
Set the url name of the endpoint instead of generating one

global_state = None
Dictionary for overiding particular values in the state

endpoint_class = None
A slug identifying the primary role of the endpoint

endpoint_classes = []
A list of slugs identifying the various roles or functions of the endpoint

api_request = None
The api request responsible for the endpoint. Generated automatically.

post_register()

get_logger()

get_site()

set_site(*site*)

site

get_parent()

set_parent(*parent*)

parent

get_endpoint_kwargs(***kwargs*)
Consult for creating child endpoints

get_common_state()

common_state

get_state()

set_state(*state*)

state
The state responsible for the endpoint. Generated automatically.

get_meta()

Return type dict

get_endpoint_classes()
Returns a list of functional identifiers

Return type list of strings

get_state_data()

Return type dict

get_state_kwargs()

Return type dict

get_state_class()

initialize_state(***data*)

reverse(**args*, ***kwargs*)
URL Reverse the given arguments

Return type string

get_base_url_name_suffix()

get_base_url_name_prefix()

get_base_url_name()
Returns the base url name to be used by our name and the name of our children endpoints Return self.base_url_name if set otherwise return the concat of self.get_base_url_name_prefix() and self.get_base_url_name_suffix()

get_name_suffix()

get_url_name()
Returns the url name that will address this endpoint Return self.url_name is set otherwise return the result of concatting self.get_base_url_name() and self.get_name_suffx()

get_url(***kwargs*)

get_url_suffix()

create_link_collection()
Returns an instantiated LinkCollection object

Return type LinkCollection

get_link_prototypes()
return a list of tuples containing link prototype class and kwargs

register_link_prototypes()

create_link_prototypes()
Instantiates the link prototypes from get_link_prototypes

Return type list of link prototypes

get_link_prototype_kwargs(***kwargs*)

Return type dict

create_link_prototype(*klass*, ***kwargs*)

fork(***kwargs*)

Return type endpoint

fork_state(***kwargs*)

Return type endpoint

get_resource_item_class()

get_resource_item(*instance*, ***kwargs*)
Wraps an object in a resource item

Return type resource item

get_instances()
Returns the list of active objects available for this request

Return type list of objects

get_resource_items()
Returns a list of resource items available for this request. Calls get_instances for the objects the items represent.

Return type list of resource items

get_form_class()

get_form_kwargs(***kwargs*)

get_item_form_class()

get_item_form_kwargs(*item=None*, ***kwargs*)

Return type dict

get_namespaces()

Return type dictionary of namespaces

get_item_namespaces(*item*)

Parameters **item** – resource item

Return type dictionary of namespaces

get_item_url(*item*)

get_item_link(*item*, ***kwargs*)

get_main_link_name()

get_main_link_prototype()

get_link(***kwargs*)

get_item_prompt(*item*)
Returns a string representing the resource item

get_prompt()
Returns a string representing this endpoint

api_permission_check(*api_request*, *endpoint*)

generate_response(*link*)

generate_options_response(*links*)

create_internal_apirequest(***kwargs*)

create_apirequest(***kwargs*)

expand_template_names(*suffixes*)

get_context_data(***kwargs*)

generate_api_response(*api_request*)

Return type Link or HttpResponse

emit_event(*event*, *item_list=None*)
Fires of the *endpoint_event* signal

get_datatap(*instream=None*, ***kwargs*)
Returns a datatap that can serialize hypermedia items and deserialize to native instances

Parameters **instream** – A list of resource items or a primitive datatap

get_native_datatap_instream_from_items(*items*)
Makes an instream of item forms

get_native_datatap(*instream=None*, ***kwargs*)
Returns a datatap that can serialize the forms belonging to hypermedia items.

VirtualEndpoint

class hyperadmin.endpoints.**VirtualEndpoint**(***kwargs*)
A type of endpoint that does not define any active endpoints itself but references other endpoints.

name_suffix = 'virtual'

get_children_endpoints()

get_urls()

get_extra_urls()

urls

dynamic_urls()

urlpatterns

get_url_object()

create_link_prototypes()
Inludes the link prototypes created by the children endpoints

get_index_endpoint()

get_main_link_name()

get_url(***kwargs*)

generate_api_response(*api_request*)
Calls the index endpoint and returns it's api response :rtype: Link or HttpResponse

RootEndpoint

class hyperadmin.endpoints.**RootEndpoint**(***kwargs*)
Bases: hyperadmin.endpoints.APIRequestBuilder, hyperadmin.endpoints.VirtualEndpoint

The top endpoint of a hypermedia aware site

Child endpoints bind to this and this endpoint is used to mount in urls.py

namespace = None
The namespace of this endpoint, will be autogenerated if none is supplied

media_types = None
Dictionary of supported media types

template_paths = None
List of template paths to use for template name resolution

base_url_name = ''

name_suffix = 'virtualroot'

link_prototypes

parent

get_logger()

post_register()

get_site()

site

fork(***kwargs*)

urls

reverse(*name*, **args*, ***kwargs*)

get_resolver()

call_endpoint(*url*, ***request_params*)
Looks up the endpoint as an internal api request :rtype: Bound Endpoint

register_media_type(*media_type*, *media_type_handler*)

record_endpoint(*endpoint*, *url_name=None*)

get_endpoint_from_urlname(*urlname*)

api_permission_check(*api_request*, *endpoint*)
Return a link describing the authentication failure or return None if the request has sufficient permissions

get_template_paths()

expand_template_names(*suffixes*)
Maps a list of template names to the template paths belonging to the site :param suffixes: List of strings :rtype: List of strings

get_context_data(***kwargs*)

Endpoint

class hyperadmin.endpoints.**Endpoint**(***kwargs*)
Bases: hyperadmin.endpoints.GlobalSiteMixin, hyperadmin.views.EndpointViewMixin, hyperadmin.endpoints.BaseEndpoint

Endpoint class that contains link prototypes and maps HTTP requests to those links.

url_suffix = None

base_url_name_suffix = ''

prototype_method_map = {}

get_available_methods()

get_link_prototype_for_method(*method*)
Return the link prototype representing the action for the method Consults prototype_method_map for the link name and returns the prototype from link_prototypes

get_link_kwargs(***kwargs*)

get_available_links()
Returns a dictionary mapping available HTTP methods to a link

get_url_suffix()

get_view_kwargs(***kwargs*)

Return type dict

get_view(***kwargs*)

Return type view callable

get_url_object()

get_main_link_name()

1.8.3 API Requests

APIRequest

class hyperadmin.apirequests.**APIRequest**(*site*, *path*, *url_args*, *url_kwargs*, *global_state=None*)
An API Request

get_django_request()

META

media_types

get_response_type()
Returns the active response type to be used

Return type string

get_request_type()
Returns the active request type to be used

Return type string

get_request_media_type()
Returns the request media type to be used or raises an error

Raises ValueError when the requested content type is unrecognized

Return type string

get_response_media_type()
Returns the response media type to be used or raises an error

Raises ValueError when the requested content type is unrecognized

Return type string

get_endpoint(*urlname*)
Returns a bound endpoint matching the urlname

Parameters **urlname** (*string*) – The urlname to find

Raises KeyError when the urlname does not match any endpoints

Return type Endpoint

record_endpoint(*endpoint*)
Record the endpoint in our urlname cache

Parameters **resource** – Endpoint

get_link_prototypes(*endpoint*)
Returns the link prototypes to be used by the endpint

Parameters **endpoint** – endpoint object

Return type list of link prototypes

get_site()
Returns the bound site

Return type SiteResource

generate_response(*link*, *state*)
Returns a response generated from the response media type

Parameters

- **link** – The active link representing the endpoint's response
- **state** – The endpoint's state

Return type [Http]Response

generate_options_response(*links*, *state*)
Returns an OPTIONS response generated from the response media type

Parameters

- **links** – dictionary mapping available HTTP methods to a link
- **state** – The endpoint's state

Return type [Http]Response

reverse(*name*, **args*, ***kwargs*)

HTTPAPIRequest

class hyperadmin.apirequests.**HTTPAPIRequest**(*request*, ***kwargs*)
Bases: hyperadmin.apirequests.APIRequest

Represents an API Request spawned from a Django HTTP Request

get_to_meta_map = {'_CONTENT_TYPE': 'CONTENT_TYPE', '_HTTP_ACCEPT': 'HTTP_ACCEPT'}

payload

method

get_django_request()

get_full_path()

user

params

get_session_data_from_request(*request*)

populate_session_data_from_request(*request*)

patched_meta(*request*)

get_passthrough_params(*request*)

NamespaceAPIRequest

class hyperadmin.apirequests.**NamespaceAPIRequest**(*api_request*, ***kwargs*)
Bases: hyperadmin.apirequests.InternalAPIRequest

user

get_django_request()

Namespace

class hyperadmin.apirequests.**Namespace**(*name*, *endpoint*, *state_data={}*)
Bases: object

Represents alternative data associated with the current api request

Namespaced data is provided by another resource through an internal api request

get_namespaces()

get_prompt()

link

state

1.8.4 Links

Links

class hyperadmin.links.**Link**(*url*, *endpoint*, *method='GET'*, *prompt=None*, *description=None*, *form=None*, *form_class=None*, *form_kwargs=None*, *on_submit=None*, *errors=None*, *link_factor=None*, *include_form_params_in_url=False*, *mimetype=None*, *descriptors=None*, *template_name='link.html'*, *cu_headers=None*, *cr_headers=None*, ***cl_headers*)

A link in the broad hypermedia sense

api_request

resource

state

site

rel

classes

get_base_url()

clone_into_links()

get_absolute_url()

The url for this link

mimetype_is_audio()

mimetype_is_image()

mimetype_is_video()

get_link_factor()

Returns a two character representation of the link factor.

- LI - Idempotent
- LN - Non-Idempotent
- LT - Templated link
- LO - Outbound link
- LI - Embedded link

is_simple_link

Returns True if this link is simply to be followed

method

The HTTP method of the link

class_attr()

get_form_kwargs(***form_kwargs*)

get_form(***form_kwargs*)

form

Returns the active form for the link. Returns None if there is no form.

errors

Returns the validation errors belonging to the form

submit(***kwargs*)
Returns a link representing the result of the action taken. The resource_item of the link may represent the updated/created object or in the case of a collection resource item you get access to the filter items

follow()
Follows the link to the endpoint in a subrequest Returns a link representing the endpoint response

clone(***kwargs*)

get_context_data(***kwargs*)

get_template_names()

expand_template_names(*suffixes*)

get_context_instance()

render(***kwargs*)

LinkCollection

class hyperadmin.links.**LinkCollection**(*endpoint*)

link_prototypes

add_link(*link_name*, ***kwargs*)
Adds the specified link from the resource. This will only add the link if it exists and the person is allowed to view it.

LinkPrototype

class hyperadmin.links.**LinkPrototype**(*endpoint*, *name*, *link_kwargs={}*)
Incapsulates logic related to a link. This class is responsible for:

- creating link
- handling link submission
- controlling link visibility

resource

state

common_state

api_request

show_link(***kwargs*)
Checks the state and returns False if the link is not active.

Return type boolean

get_link_description()

get_form_class()

get_form_kwargs(***kwargs*)

Return type dict

get_link_kwargs(***kwargs*)

Return type dict

get_link_class()

get_link(***link_kwargs*)
Creates and returns the link

Return type Link

handle_submission(*link*, *submit_kwargs*)
Called when the link is submitted. Returns a link representing the response.

Return type Link

on_success(*item=None*)
Returns a link for a successful submission

Return type Link

get_url(***kwargs*)

get_url_name()

1.8.5 Hyperobjects

These are objects generated by the resource and are serialized by a media type.

Item

class hyperadmin.hyperobjects.**Item**(*endpoint*, *instance*, *datatap=None*)
Represents an instance that is bound to an endpoint

form_class = None

link_collector_class
alias of ItemLinkCollectionProvider

state

get_absolute_url()

get_form_class()

get_form_kwargs(***kwargs*)

get_form(***form_kwargs*)

form
Mediatype uses this form to serialize the result

get_prompt()
Returns a string representing the item

get_resource_items()

get_namespaces()
Returns namespaces associated with this item

get_link(***kwargs*)

get_outbound_link(***kwargs*)

1.8.6 Indexes

class hyperadmin.indexes.**Index**(*name*, *resource*)
Encapsulates logic for doing lookups & filters on a resource

- provide links for filtering
- method for item lookup
- url params for item lookup

paginator_class = None

page_var = 'p'

state

register_filter(*a_filter*, ***kwargs*)

populate_state()

get_index_query()

get_url_params(*param_map={}*)
returns url parts for use in the url regexp for conducting item lookups

get_url_params_from_item(*item*, *param_map={}*)

get(***kwargs*)

get_resource_item(***kwargs*)

get_filtered_index()

get_link(***kwargs*)

get_filter_links(***link_kwargs*)

get_paginator_kwargs()

get_paginator(***kwargs*)

get_pagination_links(***link_kwargs*)

get_advaned_link()
Return a link with all the options in one form, ignores pagination

get_links(***kwargs*)

get_page()

class hyperadmin.indexes.**PrimaryIndex**(*name*, *resource*)
Bases: hyperadmin.indexes.Index

get_paginator_kwargs()

1.8.7 Filters

class hyperadmin.filters.**BaseFilter**(*index*)

title = None

state

make_link(***kwargs*)

populate_state()

get_links(***link_kwargs*)
Returns links representing the filterable actions.

filter_index(*active_index*)
Returns the filtered queryset.

expected_parameters()
Returns the list of parameter names that are expected from the request's query string and that will be used by this filter.

is_active()

values()

class hyperadmin.filters.**BaseChoicesFilter**(*index*)
Bases: hyperadmin.filters.BaseFilter

get_links(***link_kwargs*)

choices()

class hyperadmin.filters.**SimpleFilter**(*index*)
Bases: hyperadmin.filters.BaseChoicesFilter

parameter_name = None

has_output()

value()
Returns the value (in string format) provided in the request's query string for this filter, if any. If the value wasn't provided then returns None.

lookups()
Must be overriden to return a list of tuples (value, verbose value)

expected_parameters()

choices()

1.8.8 Media types

MediaType Interface

class hyperadmin.mediatypes.common.**MediaType**(*api_request*)

recognized_media_types = []

classmethod register_with_builtins()

site

get_django_request()

handle_redirect(*link*, *content_type*)

detect_redirect(*link*)

serialize(*content_type*, *link*, *state*)
Return an HttpResponse

options_serialize(*content_type*, *links*, *state*)
Return an HttpResponse describing the available OPTIONS at an endpoint

Parameters **links** – dictionary mapping available HTTP methods to a link

deserialize()
returns keyword arguments for instantiating a form

prepare_field_value(*val*)

get_form_instance_values(*form*, *include_initial=True*)

get_related_resource_from_field(*field*)

get_html_type_from_field(*field*)

1.8.9 Built-in Media types

JSON

class hyperadmin.mediatypes.json.**JSON**(*api_request*, ***kwargs*)
Bases: hyperadmin.mediatypes.datatap.DataTap

recognized_media_types = ['application/json']

class hyperadmin.mediatypes.json.**JSONP**(*api_request*, ***kwargs*)
Bases: hyperadmin.mediatypes.json.JSON

recognized_media_types = ['text/javascript']

get_jsonp_callback()

serialize(*content_type*, *link*, *state*)

HTML

class hyperadmin.mediatypes.html5.**Html5MediaType**(*api_request*)
Bases: hyperadmin.mediatypes.common.MediaType

template_name = 'hyperadmin/html5/resource.html'

template_dir_name = 'hyperadmin/html5'

response_class
alias of TemplateResponse

recognized_media_types = ['text/html', 'text/plain', 'application/xhtml+xml', 'application/text-html', 'application/x

get_context_data(*link*, *state*)

get_template_names(*state*)

serialize(*content_type*, *link*, *state*)

get_option_template_names()

options_serialize(*content_type*, *links*, *state*)

deserialize()

check_csrf(*request*)

Collection JSON

class hyperadmin.mediatypes.collectionjson.**CollectionJSON**(*api_request*)
Bases: hyperadmin.mediatypes.common.MediaType

recognized_media_types = ['application/vnd.Collection+JSON', 'application/vnd.collection+json']

prepare_field_value(*val*)

convert_field(*field*)

links_for_item(*item*)

convert_item(*item*)

convert_form(*form*)

convert_link(*link*)

convert_errors(*errors*)

prepare_collection(*form_link*, *state*)

prepare_link(*form_link*)

serialize(*content_type*, *link*, *state*)

options_serialize(*content_type*, *links*, *state*)

deserialize()

class hyperadmin.mediatypes.collectionjson.**CollectionNextJSON**(*api_request*)
Bases: hyperadmin.mediatypes.collectionjson.CollectionJSON

recognized_media_types = ['application/vnd.Collection.next+JSON']

convert_field(*field*)

convert_errors(*errors*)

convert_link(*link*)

class hyperadmin.mediatypes.collectionjson.**CollectionHyperAdminJSON**(*api_request*)
Bases: hyperadmin.mediatypes.collectionjson.CollectionNextJSON

recognized_media_types = ['application/vnd.Collection.hyperadmin+JSON']

get_accepted_namespaces()

convert_field(*field*)

prepare_collection(*form_link*, *state*, *include_namespaces=True*)

iFrame

class hyperadmin.mediatypes.iframe.**IframeMediaType**(*api_request*)
Bases: hyperadmin.mediatypes.common.MediaType

template_name = 'hyperadmin/iframe/resource.html'

response_class
alias of TemplateResponse

recognized_media_types = ['text/html-iframe-transport;level=1']

get_response_type()

get_response_media_type()

get_context_data(*link*, *state*)

get_template_names()

serialize(*content_type*, *link*, *state*)

deserialize()

check_csrf(*request*)

1.8.10 States

State

class hyperadmin.states.**State**(*substates*=[ ], *data={}*)

get_dictionaries()

pop(*key*, *default=None*)

update(*other_dict*)

EndpointState

class hyperadmin.states.**EndpointState**(*endpoint*, *meta*, *substates*=[ ], *data={}*)
Bases: hyperadmin.links.LinkCollectorMixin, hyperadmin.states.State

Used by resources to determine what links and items are available in the response.

link_collector_class
alias of EndpointStateLinkCollectionProvider

api_request

get_link_collector_kwargs(***kwargs*)

get_dictionaries()

resource

site

reverse(*name*, **args*, ***kwargs*)

item

meta

get_link_url(*link*)

has_view_class(*cls*)

params
The filter and pagination parameters

namespace

get_resource_items()
Returns resource items that are associated with this state.

get_query_string(*new_params=None*, *remove=None*)

get_namespaces()

1.8.11 Sites

BaseResourceSite

class hyperadmin.sites.**BaseResourceSite**(***kwargs*)

directory_resource_class
alias of ResourceDirectory

throttle = <hyperadmin.throttle.Throttle object at 0x239dc90>

name = 'hyperadmin'

post_register()

get_directory_resource_kwargs(***kwargs*)

create_directory_resource(***kwargs*)

get_children_endpoints()

get_index_endpoint()

get_resource_kwargs(***kwargs*)

get_endpoint_kwargs(***kwargs*)

fork(***kwargs*)

register_endpoint(*klass*, ***options*)

register_builtin_media_types()

get_html_type_from_field(*field*)

get_media_resource_urlname()

get_media_resource()

get_related_resource_from_field(*field*)

api_permission_check(*api_request*, *endpoint*)

ResourceSite

class hyperadmin.sites.**ResourceSite**(***kwargs*)
A Resource Site that is suited for administrative purposes. By default the user must be a staff user.

auth_resource_class
alias of AuthResource

name = 'hyperadmin'

base_url_name_suffix = 'admin'

post_register()

applications

register(*model_or_iterable*, *admin_class*, ***options*)

register_application(*app_name*, *app_class=None*, ***options*)

get_login_link(*api_request*, ***kwargs*)

api_permission_check(*api_request*, *endpoint*)

get_actions(*request*)

install_models_from_site(*site*)

install_storage_resources(*media_resource_class=None*, *static_resource_class=None*)

GlobalSite

class hyperadmin.sites.**GlobalSite**(***kwargs*)

A Resource Site that is meant for globally registering endpoints without needing to explicitly create a Resource Site.

name = 'apisite'

get_resolver()

1.8.12 Throttle

BaseThrottle

class hyperadmin.throttle.**BaseThrottle**

throttle_check(*api_request*, *endpoint*)

Throttle

class hyperadmin.throttle.**Throttle**(*throttle_at=150*, *timeframe=3600*, *expiration=None*)

throttle_check(*api_request*, *endpoint*)

Returns a link if the request should be throttled

get_identifier(*api_request*, *endpoint*)

user_id(*api_request*)

1.8.13 Signals

NOTE: the construction of the sender is based on the urlname and the event. This allows for listeners to be registered independently of resource construction.

endpoint_event

hyperadmin.signals.**endpoint_event**(*providing_args=None*)

Sent by the endpoint when an event occurs

Parameters

- **sender** – The full url name of the endpoint + ! + the event
- **endpoint** – The endpoint emitting the event

- **event** – A string representing the event
- **item_list** – An item list for which the event applies, may be empty

resource_event

hyperadmin.signals.**resource_event**(*providing_args=None*)
Sent by the resource when an event occurs

Parameters

- **sender** – The full url name of the resource + ! + the event
- **resource** – The resource emitting the event
- **event** – A string representing the event
- **item_list** – An item list for which the event applies, may be empty

1.9 Contributing

Hyperadmin is open-source and is here to serve the community by the community.

1.9.1 Philosophy

- **Hyperadmin is BSD-licensed. All contributed code must be either**
  - the original work of the author contributed under the BSD license
  - work taken form another BSD-compatible license
- GPL or proprietary works are not eligible for contribution
- Master branch represents the current stable release
- Develop branch represents patches accepted for the next release

1.9.2 Reporting Issues

- Issues are tracked on github.
- Please ensure that a ticket hasn't already been submitted before submitting a ticket.
- **Tickets should include:**
  - A description of the problem
  - How to recreate the bug
  - Version numbers of the relevant packages in your Django stack
  - A pull request with a unit test exemplifying the failure is a plus

1.9.3 Contributing Code

1. Fork it on github
2. Branch from the release and push to your github
3. Apply the changes you need
4. Add yourself to AUTHORS.rst
5. Once done, send a pull request (to the develop branch)

Your pull request / patch should be:

- clear
- works across all supported versions
- follows the existing code style (mostly PEP-8)
- comments included where needed
- test case if it is to fix a bug
- if it changes functionality then include documentation

1.10 Release Notes

1.10.1 0.10.0

- Added endpoint method, internal_dispatch, for quick internal api calls
- Added *hyperadmin.get_api*
- Added resource slug
- Added signals & events
- Added exception: LinkNotAvailable
- Added HyperadminJSONENcoder
- Added Detail Link, used when user does not have edit permissions
- Preserve requested content type on response
- Standardized CRUD verbage
- Fixed handling of negative primary keys
- Django 1.5 support
- Integrated with django-datataps
- Added Base64 file upload

1.10.2 0.9.1

- Fixed handling of negative primary keys (#2)
- improved order of operations in get link kwargs

1.10.3 0.9.0

- Added *hyperadmin.contrib.apikey* for key based authentication
- Allow endpoints to be directly mounted in urls with global site object
- Added *Wizard* resource for wizard workflows
- URL names are now fully dynamic
- Endpoints may now specify template and context for html rendering
- Added throttling

1.10.4 0.8.2

- Added Generic Inline Support (requires django.contrib.contenttypes)
- Seperated Autoload functionality
- Allow for links to be followed internally
- Simplified setting resource name and appplication_name

1.10.5 0.8.1

- Fixed non integer based urls for models
- Added OPTIONS method
- Autoload ModelAdmin.fieldsets

CHAPTER

TWO

HELP & FEEDBACK

We have a mailing list for general discussion and help: http://groups.google.com/group/django-hyperadmin/

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

h