

Building large app on top of Django's admin

Story about Ralph 3

Mateusz Kurek



Slides: https://mkurek.com/django_admin.pdf



#1 e-commerce internet platform in Poland



Reach



20 MM+
user accounts



1,7 MM+
sellers



100 K+
professional
sellers



90%
brand
recognition

Activity



70 MM
offers
monthly



19.9 MM
transactions
monthly



224 MM
visits
monthly



3,5 BN
PV
monthly

Allegro.tech in numbers

- 700+ microservices
- 25k+ rps on data bus
- 4.5+ PB of data
- 5.5k+ data center assets in 2 Data Centers
- 10k+ hosts (physical, virtual, cloud)

What is Ralph?

- Assets Management System for Data Center and Back Office
- fully open-sourced → <https://github.com/allegro/ralph>
- developed in Allegro since 2011
- based on Django from the very beginning

Cons of Ralph 2

- poor API, not-so-well models
- information in 2-3 subsystems
- adding new features took too long time
- two inconsistent UI - custom and admin
- "*data in admin is more reliable than in UI*"
- written in Python2

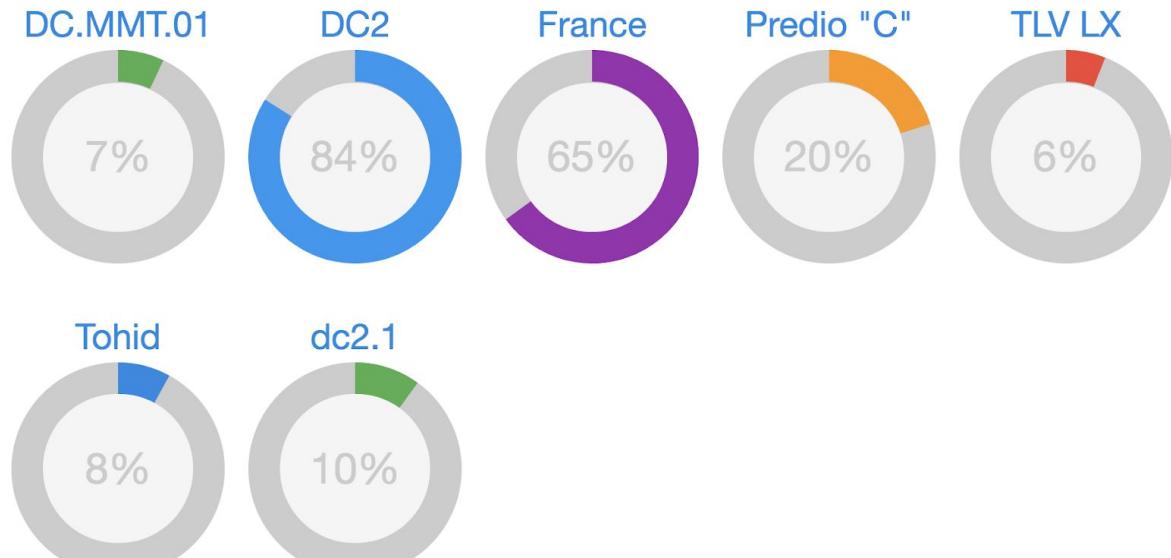
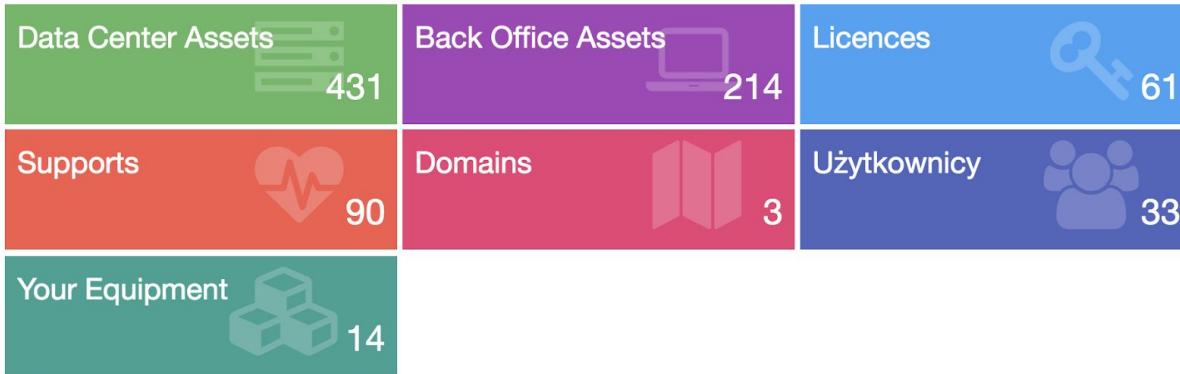
The screenshot shows the 'Add devices' page in the Ralph 2 web interface. The top navigation bar includes links for Data center, BackOffice, Licences, User list, and Supports, along with a search bar. The main content area is divided into several sections:

- DATA CENTER ACTIONS:** Includes buttons for Add device (highlighted in blue), Add part, Search, XLS upload, and Admin.
- Add devices:** A large central form for entering device details.
 - Basic Info:** Fields include Type (data center), Model (3310 NAS Gateway (Dnsstor) (Storage)), Inventory number (12354), Warehouse (North Warehouse), Location (Office tower 1), Status (In progress), Task url, Additional remarks (Here you can enter notes), Service name (Call Center), Property of (Test Property), and Hostname.
 - Financial Info:** Fields include Order no (12), Invoice date (2014-07-16), Invoice no (135/16), Price (666.66), Provider (Dnsstor), Deprecation rate (25), Source (shipment), Request date (2014-05-27), Provider order date (2014-07-14), Delivery date (2014-07-16), Deprecation end date (2015-08-06), and Budget info (new startup).
 - User Info:** Fields include User (Ralph Raphovsky IT), Owner (Ralph Raphovsky IT), Employee ID (666-999), Company (Simple Company), Department (IT), Manager (Roger Rogenick), Profit center (12), and Cost center (22).
 - Additional Info:** Fields include U level (33), U height (1), and Ralph device id.
- SN/SNs:** A section for entering serial numbers.
- Barcode/Barcodes:** A section for barcode management.

A 'Save' button is located at the bottom right of the main form area.

Let's rewrite it!

- may 2015 - decision about rewriting Ralph from scratch
- use Django's admin as the only one UI



My Recent Actions

-  test-host01 (BC: - / SN: 0) Data center asset 6 godzin, 18 minut ago
-  Rack dc2.1_a_1 (dc2.1/Server Room DC 2.1) Rack 9 godzin, 9 minut ago
-  MyRack01 (dc2.1/Server Room DC 2.1) Rack 9 godzin, 9 minut ago
-  MyRack01 - patch_panel_utp Rack accessory 9 godzin, 11 minut ago
-  MyRack01 - organizer Rack accessory 9 godzin, 11 minut ago
-  MyRack01 - patch_panel_utp Rack accessory 9 godzin, 11 minut ago
-  MyRack01 - organizer Rack accessory 9 godzin, 11 minut ago
-  MyRack01 - Switch Rack accessory 9 godzin, 12 minut ago
-  Server Room DC 2.1 (dc2.1) Server room 9 godzin, 13 minut ago
-  Rack dc2.1_a_1 (dc2.1/dc2.1_a) Rack 9 godzin, 15 minut ago
-  MyRack01 (dc2.1/dc2.1_a) Rack 9 godzin, 17 minut ago
-  MyRack01 (dc2.1/dc2.1_a) Rack 9 godzin, 17 minut ago
-  Knowledge Category 9 godzin, 18 minut ago
-  Rack dc2.1_a_1 (dc2.1/dc2.1_a) Rack 9 godzin, 19 minut ago
-  Rack dc2.1_a_1 (dc2.1/dc2.1_a) Rack 9 godzin, 19 minut ago
-  ralph257.allegro.pl (BC: dc100000257 / SN: 642-73-0773) Data center asset 9 godzin, 37 minut ago
-  c0227 (BC: bo100000227 / SN: 834-03-1772) Back Office Asset 14 godzin, 15 minut ago
-  Telephone Category 14 godzin, 46 minut ago
-  Disk broken Operation 17 godzin, 7 minut ago
-  test Custom field 17 godzin, 34 minut ago

ralph411.allegro.pl (BC: dc100000411 / SN: 790-30-0295)

Transitions

History

Basic info

Components

Network

Security Info

SCM info

Relations

Licences

Supports

Operations

Current transi...

Attachments

Basic info

Hostname: ralph411.allegro.pl

* Model: [ATS] Foxconn ML10 v21

* Status: new

Barcode: dc100000411

SN: 790-30-0295

Inventory number:

Required support

Remarks:

Tags:

Property of: Grupa Allegro SP. z o.o.

Location Info

* Rack:

Position:

* Orientation: front

Slot number:

Parent:

Management ip:

Management hostname:

Visualization: —

Usage info

* Service env: Backup systems - prod

Configuration path: order/cache

Production year:

Production use date:

Delete

Duplicate

Save

The reasons behind choosing Django admin as our foundation

- one and only one UI
- easiness of adding new domain models in declarative way
- inlines
- lot of external packages support

Built-in admin features

List of models

Django Project

WELCOME, **USER_1**. [VIEW SITE / CHANGE PASSWORD / LOG OUT](#)

Site administration

AUTH TOKEN

Tokens	+ Add	Change
--------	-----------------------	------------------------

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	Change
--------	-----------------------	------------------------

SCROOGE

Business lines	+ Add	Change
Dynamic extra costs	+ Add	Change
Environments	+ Add	Change
Extra cost types	+ Add	Change
Pricing objects	+ Add	Change
Pricing services	+ Add	Change
Services	+ Add	Change
Teams	+ Add	Change
Teams costs	+ Add	Change
Usage types	+ Add	Change
Users	+ Add	Change
Warehouses	+ Add	Change

Recent actions

My actions

- [+ gke_bytes_sent](#)
Usage type
- [✖ GCP](#)
Service
- [✖ GCP](#)
Service
- [✎ \(Dummy\)](#)
Pricing object
- [✎ foobar \(Dummy\)](#)
Pricing object
- [✎ foobar \(Unknown\)](#)
Pricing object
- [+ Google Kubernetes Engine](#)
Pricing service
- [+ gke_mem_request](#)
Usage type
- [✖ GCP CPU allocated](#)
Usage type
- [+ gke_cpu_request](#)
Usage type

List of model entities

Select user to change

ADD USER +

Search

Action: 0 of 4 selected

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	test	test@example.com			<input checked="" type="checkbox"/>
<input type="checkbox"/>	user_1	user_1@example.com			<input checked="" type="checkbox"/>
<input type="checkbox"/>	user_2	user_2@example.com			<input checked="" type="checkbox"/>
<input type="checkbox"/>	user_3	user_3@example.com			<input checked="" type="checkbox"/>

4 users

FILTER

By staff status

- All
- Yes
- No

By superuser status

- All
- Yes
- No

By active

- All
- Yes
- No

```
@admin.register(User)
class UserAdmin(admin.ModelAdmin):
    list_display = ('username', 'email', 'first_name', 'last_name', 'is_staff')
```

Simple filters and search

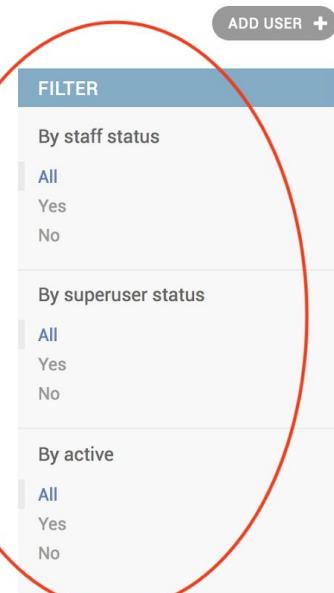
Select user to change

Action: ----- Go 0 of 4 selected

<input type="checkbox"/> USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/> test	test@example.com			<input checked="" type="checkbox"/>
<input type="checkbox"/> user_1	user_1@example.com			<input checked="" type="checkbox"/>
<input type="checkbox"/> user_2	user_2@example.com			<input checked="" type="checkbox"/>
<input type="checkbox"/> user_3	user_3@example.com			<input checked="" type="checkbox"/>

4 users

```
@admin.register(User)
class UserAdmin(admin.ModelAdmin):
    list_display = ('username', 'email', 'first_name', 'last_name', 'is_staff')
    list_filter = ('is_staff', 'is_superuser', 'is_active')
    search_fields = ('username', 'first_name', 'last_name', 'email')
```



Model form

Change Dynamic extra cost

HISTORY

Name: 

Active
If inactive, this type won't take part in costs calculation

Value rounding: Decimal places

Excluded services:

Available Service 

Filter

1234 test
another service

Chosen Service 

sample service

 Choose all  Remove all

Services excluded from cost distribution (besides usage type excluded services) Hold down "Control", or "Command" on a Mac, to select more than one.

Symbol:
(Usually) slug of the name of the usage. Used (mostly) in API to specify type of the usage.

DYNAMIC EXTRA COST DIVISIONS		PERCENT	DELETE?
USAGE TYPE	DynamicExtraCostDivision object		
<input type="text" value="transfer"/>   	<input type="text" value="100.0"/>		
<input type="text"/>   	<input type="text" value="100"/>		
<input type="text"/>   	<input type="text" value="100"/>		

 Add another Dynamic extra cost division

Delete

Save and add another

Save and continue editing

SAVE

```
@register(models.DynamicExtraCostType)
class DynamicExtraCostTypeAdmin(
    admin.ModelAdmin
):
    readonly_fields = ('symbol',)
```

But did you know that admin supports also

Actions

Select article to change ADD ARTICLE +

Action: ----- Delete selected articles Go 2 of 5 selected

	STATUS
<input type="checkbox"/> A New Human-like Species Discovered in Deep Burial Chamber	Published
<input checked="" type="checkbox"/> Django 1.9 Released	Draft
<input type="checkbox"/> Mars Is a Real Fixer-Upper of a Planet, Says Elon Musk on Colbert's 'Late Show'	Withdrawn
<input checked="" type="checkbox"/> The Coming of the Glacier Men	Draft
<input type="checkbox"/> The Last Audio Cassette Factory	Published

5 articles

```
def make_published(modeladmin, request, queryset):
    queryset.update(status='p')
make_published.short_description = "Mark selected stories as published"

class ArticleAdmin(admin.ModelAdmin):
    actions = [make_published]
```

Inlines

```
class DivisionInline(  
    admin.TabularInline  
):  
  
    model = DynamicExtraCostDivision  
  
  
@register(  
    models.DynamicExtraCostType  
)  
  
class DynamicExtraCostTypeAdmin(  
    admin.ModelAdmin  
):  
  
    inlines = [DivisionInline]
```

Change Dynamic extra cost

HISTORY

Name: test-cost

Active
If inactive, this type won't take part in costs calculation

Value rounding: 0
Decimal places

Excluded services:

Available Service Filter 1234 test another service

Chosen Service sample service

Choose all Remove all

Services excluded from cost distribution (besides usage type excluded services) Hold down "Control", or "Command" on a Mac, to select more than one.

Symbol: test-cost
(Usually) slug of the name of the usage. Used (mostly) in API to specify type of the usage.

DYNAMIC EXTRA COST DIVISIONS

USAGE TYPE	PERCENT	DELETE?
DynamicExtraCostDivision object transfer	100,0	<input type="checkbox"/>
	100	<input type="checkbox"/>
	100	<input type="checkbox"/>

Add another Dynamic extra cost division

Delete Save and add another Save and continue editing SAVE

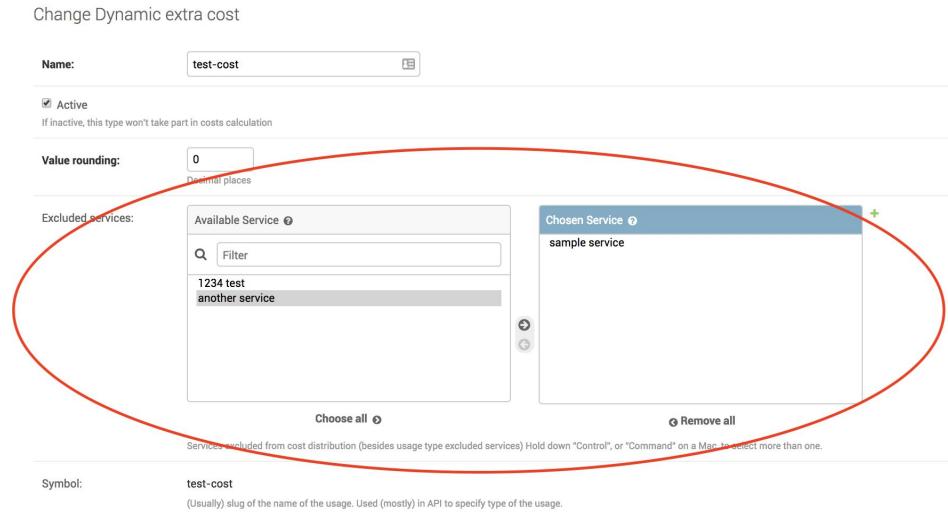


Custom form or custom widgets

```
from django import forms
from django.contrib import admin
from django.contrib.admin.widgets import (
    FilteredSelectMultiple
)

class DynamicExtraTypeForm(forms.ModelForm):
    class Meta:
        model = models.DynamicExtraCostType
        fields = '__all__'
        widgets = {
            'excluded_services': FilteredSelectMultiple(
                'Service', False
            )
        }

@register(models.DynamicExtraCostType)
class DynamicExtraCostTypeAdmin(admin.ModelAdmin):
    form = DynamicExtraTypeForm
    # ...
```



Callable field on listing

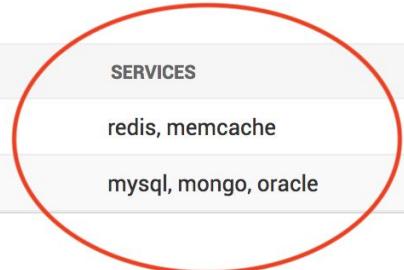
Select pricing service to change

Q Search

Action: Go 0 of 2 selected

<input type="checkbox"/>	NAME	SYMBOL	ACTIVE	PLUGIN TYPE	SERVICES
<input type="checkbox"/>	Cache	cache	<input checked="" type="checkbox"/>	universal	redis, memcache
<input type="checkbox"/>	Databases	databases	<input checked="" type="checkbox"/>	fixed price	mysql, mongo, oracle

2 pricing services



```
@register(models.PricingService)
class PricingServiceAdmin(admin.ModelAdmin):
    list_display = ('name', 'symbol', 'active', 'plugin_type', 'get_services')

    def get_services(self, pricing_service):
        return ', '.join(map(str, pricing_service.services.all()))
    get_services.short_description =_('Services')
```

And many others

- ordering
- items per page
- select related
- prefetch related

Ralph 3

How did we start?

- modified admin default listing view → [django-sitetree](#)

The screenshot shows the Ralph 3.0 web interface. At the top, there is a navigation bar with links for Data Center, Back Office, Networks, Licenses, Domains, Supports, Reports, Operations, Dashboards, and Settings. A search bar is also present. Below the navigation bar, the main content area displays the hardware details for the asset 'ralph411.allegro.pl' (BC: dc100000411 / SN: 790-30-0295). The 'Dashboards' tab is currently selected. Other tabs include Graphs, Transitions, and History.

- chosen front-end framework → foundation
- overridden a lot of Django's admin templates
- created our extension of ModelAdmin → RalphAdmin

External libraries

- [django-sitetree](#) for menu and breadcrumbs
- [django-reversion](#) for history of objects
- [django-rest-framework](#) for REST API
- [django-import-export](#) for CSV/Excel import/export of objects

Extra views

Views not related to any particular model

- create template file
- create view inheriting from Django's `TemplateView`
- add it to urls and menu

- Server rooms
 - baba
 - room1
 - DC.MMT.01
 - UC1
 - UC2
 - DC2
 - Server Room B
 - dc2.1
 - dc2.1_a
 - France
 - server room 123
 - Server Room A
 - Predio "C"
 - 3st. Andar
 - TLV LX
 - TLV Room 1A
 - Tohid
 - Mobinone

Service Catalog	-
Hostname	-
Model	-
Barcode	-
SN	-
Position	-
Height	-
Management	-
Remarks	-
-	

Extra views for particular model

- two ways: customized view (with template) or some form based on admin
- inherit from `RalphDetailView` or `RalphDetailViewAdmin`
 - automatically handle tab, icon, url etc

Test-licence-2 (BC: 100011 / SN: -)

[Basic info](#)

[Compon...](#)

[Network](#)

[Security I...](#)

[SCM info](#)

[Relations](#)

[Licences](#)

[Supports](#)

[Operations](#)

[Current tr...](#)

[Attachme...](#)

Base object licences

Licence	Quantity	Delete?
11 (8 free) x Magic - 2018-03-27 (1)	1	<input type="checkbox"/>
111 (110 free) x Photoshop - None (12345)	1	<input type="checkbox"/>
	1	

[Add another Base object licence](#)

 Basic info
 Components
 Network
 Security Info
 SCM info
 Relations
 Licences
 Supports
 Operations
 DNS
 Current tra...
 Attachments

Security scan

last scan date	2018-08-15
scan status	ok
next scan date	2018-08-22
details	-

Vulnerabilities

Name	Risk	Patch deadline
Ubuntu 14.04 LTS / 16.04 LTS / 17.10 : ubuntu-release-upgrader vulnerability (USN-3623-1)	high	2018-05-10
Ubuntu 14.04 LTS / 16.04 LTS : libxcursor vulnerability (USN-3729-1)	high	2018-09-06
Ubuntu 14.04 LTS / 16.04 LTS : gnupg vulnerability (USN-3733-1)	high	2018-09-06
Ubuntu 16.04 LTS : openjdk-8 vulnerability (USN-3734-1)	medium	2018-10-09

```
class DCAssetLicence(RalphDetailViewAdmin):  
    icon = 'key'  
    name = 'dc_asset_licences'  
    label = _('Licences')  
    url_name = 'data_center_asset_licences'  
  
    class DCAssetLicenceInline(RalphTabularInline):  
        model = BaseObjectLicence  
        raw_id_fields = ('licence',)  
        extra = 1  
  
    inlines = [DCAssetLicenceInline]
```

```
@register(DataCenterAsset)  
class DataCenterAssetAdmin(RalphAdmin):  
    # ...  
    change_views = [DCAssetLicence, DCAssetSecurityInfo]  
    # ...
```

```
class DCAssetSecurityInfo(RalphDetailView):  
    icon = 'lock'  
    label = 'Security Info'  
    name = 'security_info'  
    url_name = 'datacenter_asset_security_info'  
    template_name = 'security/security_info.html'  
  
    def get_context_data(self, **kwargs):  
        context = super().get_context_data(**kwargs)  
        try:  
            scan = self.object.securityscan  
        except ObjectDoesNotExist:  
            scan = None  
        context['security_scan'] = scan  
        return context
```

Polymorphic models

3 styles of model inheritance in Django

- Abstract base classes
- Proxy models
- Multi-table inheritance

Abstract base classes

- use the parent class to hold information that you don't want to have to type out for each child model
- database table will NOT be created for abstract model

Abstract base classes

```
class CommonInfo(models.Model):
    name = models.CharField(max_length=100)
    age = models.PositiveIntegerField()

    class Meta:
        abstract = True

class Student(CommonInfo):
    home_group = models.CharField(max_length=5)

class Employee(CommonInfo):
    company = models.CharField(max_length=50)
```

```
> .tables
abc_inheritance_employee
abc_inheritance_student
```

```
> PRAGMA table_info([abc_inheritance_student]);
cid      name       type      notnull      dflt_value      pk
-----  -----
0        id         integer   1           1
1        name        varchar(10) 1           0
2        age         integer   un 1           0
3        home_group  varchar(5)  1           0
```

```
> PRAGMA table_info([abc_inheritance_employee]);
cid      name       type      notnull      dflt_value      pk
-----  -----
0        id         integer   1           1
1        name        varchar(10) 1           0
2        age         integer   un 1           0
3        company     varchar(50) 1           0
```

Proxy models

- use it to change the Python behavior of a model (ex. default manager)
- operates on the same database table as its parent model class
- QuerySets still return the model that was requested

```
class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    age = models.PositiveIntegerField()

class OrderedPerson(Person):
    class Meta:
        proxy = True
        ordering = ["last_name"]

class UnderagePersonManager(models.Manager):
    def get_queryset(self):
        return super().get_queryset().filter(age__gte=18)

class UnderagePerson(Person):
    objects = UnderagePersonManager()
    class Meta:
        proxy = True
```

Multi-table inheritance

- each model in the hierarchy is a model all by itself
- each model corresponds to its own database table
- each model can be queried and created individually
- link between child and each of its parents model using OneToOneField

Multi-table inheritance

```
class Person(models.Model):  
    name = models.CharField(max_length=100)  
    age = models.PositiveIntegerField()  
  
class Student(Person):  
    home_group = models.CharField(  
        max_length=5  
    )  
  
class Employee(Person):  
    company = models.CharField(  
        max_length=50  
    )
```

```
> .tables  
mmi_inheritance_person  
mmi_inheritance_employee  
mmi_inheritance_student
```

```
> PRAGMA table_info([mmi_inheritance_person]);  
cid      name          type       notnull  dflt_value  pk  
----  -----  -----  -----  -----  -----  
0       id            integer    1           1  
1       name          varchar(10) 1           0  
2       age            integer    un 1           0
```

```
> PRAGMA table_info([mmi_inheritance_student]);  
cid      name          type       notnull  dflt_value  pk  
----  -----  -----  -----  -----  -----  
0       person_ptr_id  integer    1           1  
1       home_group     varchar(5)  1           0
```

```
> PRAGMA table_info([mmi_inheritance_employee]);  
cid      name          type       notnull  dflt_value  pk  
----  -----  -----  -----  -----  -----  
0       person_ptr_id  integer    1           1  
1       company        varchar(50) 1           0
```

Multi-table inheritance

```
>>> from mmi_inheritance.models import Person, Student, Employee

>>> Student.objects.create(name='John Doe', age=22, home_group='G1')
<Student: John Doe>
>>> Employee.objects.create(name='Uncle Bob', age=44, company='Allegro')
<Employee: Uncle Bob>

>>> Person.objects.all()
<QuerySet [<Person: John Doe>, <Person: Uncle Bob>]>
>>> Student.objects.all()
<QuerySet [<Student: John Doe>]>

>>> Person.objects.filter(name='Uncle Bob')
<QuerySet [<Person: Uncle Bob>]>
>>> Employee.objects.filter(name='Uncle Bob')
<QuerySet [<Employee: Uncle Bob>]>
```

Polymorphic models - goal

```
>>> Person.polymorphic_objects.all()
<QuerySet [<Student: John Doe>, <Employee: Uncle Bob>]>
```

Ralph's Polymorphic models - why?

- Common namespace of identifiers (identify object by single value) → CMDB
- Enforcement of some common fields (like service)
- Easiness of querying for related objects (ex. hosts of all types)

Ralph's Polymorphic models

```
from ralph.lib.polymorphic import Polymorphic, PolymorphicBase

class BaseObject(Polymorphic, metaclass=PolymorphicBase):
    name = models.CharField(max_length=255)
    remarks = models.TextField(blank=True)
    service = models.ForeignKey('Service')

class DataCenterAsset(BaseObject):
    model = models.ForeignKey(AssetModel)
    rack = models.ForeignKey(Rack)
    # ...

class CloudHost(BaseObject):
    cloudprovider = models.ForeignKey(CloudProvider)
    hypervisor = models.ForeignKey(DataCenterAsset)
    # ...
```

Ralph's Polymorphic models - features

- `polymorphic_select_related`, `polymorphic_prefetch_related` - allow to specify `select_related` / `prefetch_related` for child models

```
MyBaseModel.polymorphic_objects.polymorphic_select_related(  
    MyDescendantModel=['related_field1', 'related_field2'],  
    MyDescendantModel2=['related_field3'],  
)
```

- Easy access from base model to final class instance (through `last_descendant` property)
- `polymorphic_objects` is default manager

Ralph's Polymorphic models - how?

- using Django's `contenttypes` framework
- custom `QuerySet` manager with heavily rewritten `iterator`:
 - Call super's `iterator` to get list of base objects and store it's content types and primary keys
 - Group objects by content types
 - For each content type call query on its model filtering by primary keys (with polymorphic filters)
 - Restore original order of items
- optimized number of queries: $1 + \text{num of content types}$

Transitions

Transitions

- set of actions to run on some object(s)
- originally should involve transitioning from one status to another
- actions defined in code, transitions defined in UI
- might be synchronous or asynchronous
- might be in “frozen” state and wait for external action (ex. HTTP request)
- might produce attachment(s)

Deploy	<input type="checkbox"/> new in use free damaged <small>disabled</small>	Keep orginal status <input checked="" type="checkbox"/>	ASYNC_TRANSITIONS	Default <input type="button" value="▼"/>		<input type="checkbox"/> Assign additional IP and hostname pair <input checked="" type="checkbox"/> Change configuration_path <input checked="" type="checkbox"/> Assign new hostname <input type="checkbox"/> Assign new IP <input checked="" type="checkbox"/> Change service-env <input type="checkbox"/> Change rack <input checked="" type="checkbox"/> Clean DHCP entries <input type="checkbox"/> Clean DNS entries <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Clean hostname <input checked="" type="checkbox"/> Clean IP addresses <input type="checkbox"/> Cleanup scm status <input type="checkbox"/> Cleanup security scans <input type="checkbox"/> Convert to BackOffice Asset <input checked="" type="checkbox"/> Assign new IP address and create DHCP entries <input checked="" type="checkbox"/> Create DNS entries <input checked="" type="checkbox"/> Apply preboot <input checked="" type="checkbox"/> <input type="checkbox"/> Remove from DHCP entries <input type="checkbox"/> Replace IP <input checked="" type="checkbox"/> Wait for DHCP servers <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> Wait for ping <input checked="" type="checkbox"/>
--------	--	---	-------------------	--	--	---

Location Info

60c G7 2x...					

* Rack: Remove from DHCP

Position: 17 Assign new IP (without hostname)

* Orientation: from Reinstall-to-cleaned

Slot number: 6 Assign additional IP and hostname

Transitions ▾ Hist...

Data center assets

Actions

✓ -----

- Delete selected data center assets
- Invoice report
- Bulk edit
- Convert to backoffice asset transition
- Change config path transition
- Deploy transition**
- Reinstall transition
- Remove from dhcp transition
- Assign new ip (without hostname) transition
- Cleanup transition
- Change main hostname transition
- Reinstall-to-cleaned transition
- Assign additional ip and hostname transition

Configuration class path

Configuration class module

MAC Address

<input type="checkbox"/> Hostname	Status
<input checked="" type="checkbox"/> s12345.mydc.net	in use
<input checked="" type="checkbox"/> s54321.mydc.net	in use
<input checked="" type="checkbox"/> s123.anotherdc.net	in use

Transition deploy for data center asset:

- s12345.mydc.net

Option	Value
Hostname:	s54321.mydc.net
IP Address:	10.20.30.40 (prod-net.mydc.net (10.20.30.0/24 VLAN: 201))
MAC Address:	AA:BB:11:22:33:FF
Service-environment:	Ralph - test ? ✖ 🔍 ℹ
Configuration path:	ralph_test/www ✖ 🔍 ℹ
Preboot:	Ubuntu 16.04

Cancel
transition

Run transition

s12345.mydc.net

Transition history

Date	Author	Transition	Source	Target	Actions	Data	Attachment
Jan. 31, 2018, 8:37 a.m.	John Doe (john.doe)	Release asset	used	used	Assign owner Assign user Assign warehouse Release report	warehouse: PL POZ owner: Mateusz Kurek (mateusz.kurek) user: Mateusz Kurek (mateusz.kurek)	1. 2018-01-31_mateusz-kurek_release.pdf
Jan. 29, 2018, 3:53 p.m.	Uncle Bob (uncle.bob)	Deploy	waiting_for_release	waiting_for_release	Assign hostname if empty or country not match Assign licence Assign owner Assign user	licences: owner: Mateusz Kurek (mateusz.kurek) user: Mateusz Kurek (mateusz.kurek)	—

Transition history

Date	Author	Transition	Source	Target	Actions	Data	Attachment
July 26, 2018, 11:49 a.m.	Mateusz Kurek (mateusz.kurek)	Reinstall	used	—	Apply preboot Wait for ping	Preboot: Ubuntu 16.04	—
July 26, 2018, 10:16 a.m.	Mateusz Kurek (mateusz.kurek)	Reinstall	used	—	Apply preboot Wait for ping	Preboot: Ubuntu 16.04	—
Oct. 5, 2017, 7:14 a.m.	Mateusz Kurek (mateusz.kurek)	Remove from DHCP	used	—	Remove from DHCP entries	DHCP entry: 10.20.30.40 (s12345.mydc.net) / AA:BB:11:22:33:FF	—
Oct. 4, 2017, 4:07 p.m.	Mateusz Kurek (mateusz.kurek)	Reinstall	used	—	Apply preboot Wait for ping	Preboot: Ubuntu 16.04	—

Transitions - how?

- lot of plain Django stuff - forms, models, views
- create transition-related objects in post-migration step
- asynchronous transitions done using [RQ](#)
- `transition_action` decorator for configuring transition actions

Transitions - sample actions

```
from ralph.lib.transitions import transition_action

class DataCenterAsset(models.Model):
    @classmethod
    @transition_action(
        verbose_name=_('Change rack'),
        form_fields={'rack': {
            'field': forms.CharField(
                widget=AutocompleteWidget(
                    field=rack, admin_site=ralph_site
                )
            ),
        }},
    )
    def change_rack(cls, instances, **kwargs):
        rack = Rack.objects.get(pk=kwargs['rack'])
        for instance in instances:
            instance.rack = rack
```

```
class BackOfficeAsset(models.Model):
    @classmethod
    @transition_action(
        form_fields={'language': {
            'field': forms.ModelChoiceField(
                label=_('Return report language'),
                queryset=ReportLanguage.objects.all()
            ),
            'exclude_from_history': True
        }},
        return_attachment=True,
        precondition=_check_user_assigned,
    )
    def return_report(cls, instances, requester, **kwargs):
        return cls._generate_report(
            instances=instances, name='return',
            requester=requester, language=kwargs['language']
        )
```

Granular permissions

Permissions in Django

- Django Admin has add, change and delete permission for model
- Django lack of view permission (out-of-the-box) before Django 2.1
- lot of open-source packages:
 - [django-guardian](#)
 - [django-rules](#)

4 levels of permissions

- permission for model
- permission for object
- permission per field
- permission for extra tabs and transitions

* Name: some_group



Permissions:

Accessory	1 of 4
Asset	25 of 54
Asset Holder	1 of 4
Attachment	1 of 4
Attachment Item	1 of 4
Back Office Asset	1 of 95
Base Object	9 of 30
Base Object Cluster	0 of 4
Base Object Licence	2 of 4
Base Objects Support	2 of 4
Budget Info	1 of 4
Business Segment	1 of 4
Category	1 of 4
Change	0 of 4
Cloud Flavor	13 of 24
Cloud Host	18 of 33
Cloud Project	12 of 24
Cloud Provider	1 of 4

Accessory

0 of 4

- All
- Can add accessory
- Can change accessory
- Can delete accessory
- Can view accessory

Data Center Asset

51 of 104

- All
- Can add data center asset
- Can run assign additional ip and hostname transition
- Can run assign new ip (without hostname) transition
- Can run change config path transition
- Can run change main hostname transition
- Can run cleanup transition
- Can run convert to backoffice asset transition
- Can run deploy transition
- Can run reinstall transition
- Can run reinstall-to-cleaned transition
- Can run remove from dhcp transition
- Can view DataCenterAssetAdminAttachmentsView
- Can view DataCenterAssetAdminCurrentTransitionsView
- Can view DataCenterAssetComponents
- Can view DataCenterAssetLicence
- Can view DataCenterAssetNetworkView
- Can view DataCenterAssetOperation
- Can view DataCenterAssetRelationsView
- Can view DataCenterAssetSCMInfo
- Can view DataCenterAssetSecurityInfo
- Can view DataCenterAssetSupport
- Can view DNSView

Cloud Project

11 of 24

- All
- Can add cloud project
- Can change cloud project
- Can change cloudprovider field
- Can change configuration path field
- Can change content type field
- Can change name field
- Can change parent field
- Can change project id field
- Can change remarks field
- Can change service env field
- Can change Tags field
- Can delete cloud project
- Can view cloud project
- Can view baseobject ptr field
- Can view cloudprovider field
- Can view configuration path field
- Can view content type field
- Can view ID field
- Can view name field
- Can view parent field
- Can view project id field
- Can view remarks field
- Can view service env field
- Can view Tags field

Permission for model

- scan all models and all its fields etc. and create permissions for them
- permission for adding/changing/deleting model is handled by Django
- render menu for each user basing on their permissions

Ralph 3

Back Office

Licenses

Supports

Settings

Back Office Assets



22698

Licen

Users



15384

Asset model

Service Environment

Asset holder

Users list

Regions

Custom fields

Office Infrastructures

Supports



1329

1767



0

Ralph 3

Data Center

Networks

Reports

Operations

Settings

Data Center Assets



8513

Your Equipment

Asset model

Manufacturer

Service

Service Environment

Custom fields

Permission for object

- override some of Admin methods:
 - has_change_permission
 - has_delete_permission
 - get_queryset
- Model has to
 - inherit from PermissionsForObjectMixin
 - add Permissions inner class with has_access property
- RalphAdmin provides both permission for object and per field

```
@user_permission
def object_has_region(user):
    """
    Check if object's region is one of user regions.
    """
    return models.Q(region__in=user.regions_ids)

class BackOfficeAsset(PermissionsForObjectMixin):
    region = models.ForeignKey(Region, blank=False, null=False)

    class Permissions:
        has_access = object_has_region
```

Permissions per field

- the most tricky
- override Admin methods (to narrow down form to fields user has access):
 - get_fieldsets
 - get_READONLY_FIELDS
 - get_form
 - get_list_display
- model has to inherit from PermByFieldMixin abstract model

```
from ralph.lib.permissions import PermByFieldMixin

class LicenceType(PermByFieldMixin, models.Model):
    pass
```

[Basic info](#)[Assignments](#)[Assigned t...](#)[Attachments](#)**Basic info**

* Licence type: OTHER   

Manufacturer:    

* Software: Sublime Text     

* Inventory number: 94842

SN / key: XXX

Valid thru: 

License details:

* Region: PL  

Remarks:

Financial info

Order no:    

Invoice no:    

Price:    

Depreciation rate: 50,00 

Invoice date:    

* Number of purchased items: 1

Used: 1

Free: 0

Accounting id:    

Budget info:    

Provider:    

Office infrastructure:    

Property of: -----  

[Delete](#)[Save](#)

1 x Sublime Text - None (94842)

[History](#)[Basic info](#)[Assignments](#)[Assigned t...](#)[Attachments](#)

Basic info

Licence type: OTHER



Manufacturer: (None)

Software: Sublime Text

Inventory number: 94842

SN / key: XXX

Valid thru: (None)

License details:

Region: PL

Remarks:

Service env: (None)

Financial info

Order no: -

Invoice no: -

Price: (None)

Invoice date: (None)

Number of purchased items: 1

Used: 1

Free: 0

Accounting id: -

Budget info: (None)

Provider: -

Office infrastructure:



Property of: (None)

[Save](#)

i Basic info[Licence](#)[Supports](#)[Attachments](#)[Current tra...](#)**Basic info**

Hostname:



* Model: [PC] Apple MAC MINI

Barcode: 65748

SN: SN12345

Imei: (None)

Inventory number: (None)

* Status: new

* Warehouse: PL OFFICE1



Location:

Region: PL

Loan end date:

Remarks: MGGEN2MP/A

User Info

User:



Owner:

**Financial Info**

Order number: -

Purchase order: 21312323

Invoice date: (None)

Invoice number: -

Price: 0.00



Depreciation rate: 40.00

Depreciation end date: (None)



Force depreciation:

Provider: Cortland



Budget info: (None)

[Duplicate](#)[Save](#)

Custom fields

Custom fields

- allows to attach non-hardcoded attribute and value to any object
- inheritance of custom field values from parent objects (ex. service)
- Custom field's value could have one of many types (ex. string, integer, choice from list)

monitoring_prototype

* Name: i

Attribute name: monitoring_prototype i

* Type: choice list i

Choices: manual|auto|backend|frontend|not-monitored i

Default value: i

Use as configuration variable i

Custom field values

Key	Value	Delete?
i	i	
i	i	
i	i	

Backup systems - prod

History

*** Service:** Backup systems*** Environment:** prod

Remarks:

Tags:



Custom field values

Key	Value	Clear children values?	Delete?
cluster	common	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	
		<input type="checkbox"/>	
		<input type="checkbox"/>	

Custom field values

Key	Value	Delete?
monitoring_prototype     	backend	<input type="checkbox"/>
	  	
	  	
	  	

Custom field values summary

Custom field	Value	Inherited from
monitoring_prototype	backend	-
cluster	common	service environment: Backup systems - prod

Custom fields - how?

- Using content types to attach to any model/object
- Custom QuerySet, GenericRelation and GenericDescriptor
- Custom Form and admin Inline
- Additional resource in API for each model:

`http://ralph/api/<YOUR-RESOURCE-URL>/customfields/`

Custom fields - how?

```
from ralph.lib.custom_fields import (
    CustomFieldMeta,
    WithCustomFieldsMixin
)

class ConfigurationModule(
    WithCustomFieldsMixin,
    models.Model,
    metaclass=CustomFieldMeta
):
    # ...
    pass
```

```
from ralph.lib.custom_fields import CustomFieldValueAdminMixin

@register(ConfigurationModule)
class ConfigurationModuleAdmin(CustomFieldValueAdminMixin, RalphAdmin):
    show_custom_fields_values_summary = False
    # ...
```

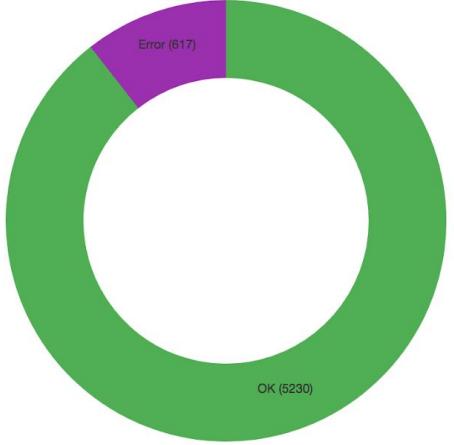
```
from ralph.lib.custom_fields.api import WithCustomFieldsSerializerMixin

class ConfigurationModuleSerializer(
    WithCustomFieldsSerializerMixin,
    RalphAPISerializer
):
    class Meta:
        model = ConfigurationModule
        fields = ('id', 'url', 'name', 'custom_fields')
```

Charts and dashboards

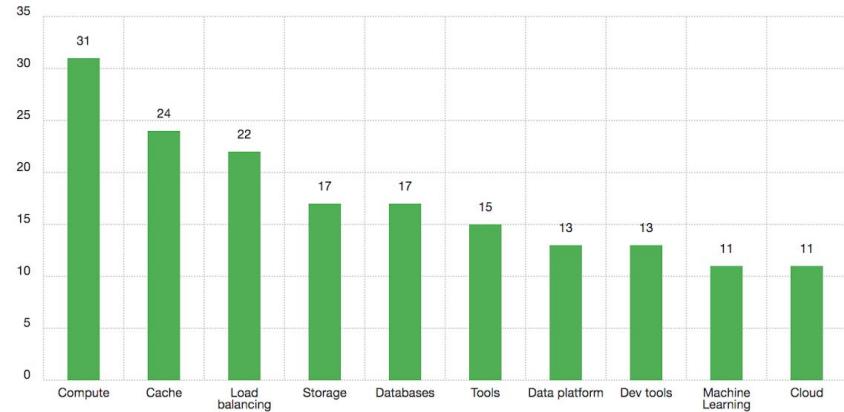
Charts and dashboards

- allow to visualize data from Ralph easily
- using ORM-like declarative query
- using [chartist](#) to display charts
- charts could be connected together in dashboards



SCM status

Failed DC Hosts SCM Status by service only prod



[assets] Workstations on stock with status "free"

* Name: [assets] Workstations on stock with status "free"

Description:

* Model: Back Office Asset

* Aggregate type: Count

* Chart type: Vertical Bar

Params:

```
{  
    "filters": {  
        "model__category__in": [  
            87,  
            88  
        ],  
        "region_id": 6,  
        "status": 10  
    },  
    "labels": "model__category__name",  
    "series": "barcode"  
}
```

Active

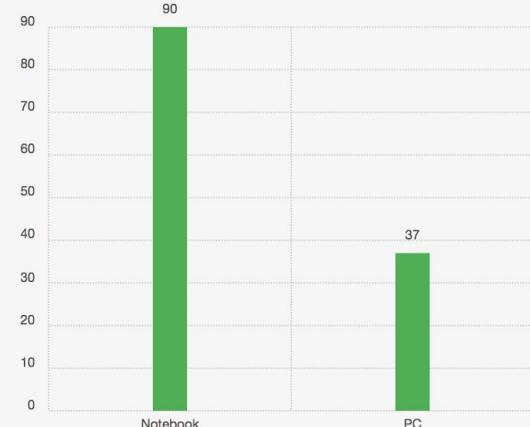
Push to statsd



Preview

Graph:

[assets] Workstations on stock with status "free"



-
- e: ✓ Count
 - e: Count with zeros
 - e: Max
 - e: Sum
 - s: Sum with zeros
 - s: Sum boolean values
 - s: Sum negated boolean values
 - s: Ratio

Assets

- model
- category
- manufacturer
- service
- service environment

Back_office

- ✓ Back Office Asset
- warehouse
- Data_center
- data center
- data center asset
- dc host
- rack
- server room

Licences

- licence
- software

Operations

- change
- failure
- incident
- operation
- problem

Security

- security scan
- vulnerability

Supports

- support

Virtual

- Cloud host
- Virtual server (VM)

Filtering

- built-in Django's Admin filters are (too) simple
- we've created our own filters for a lot of types of fields (like text field, choice field, foreign key etc)
- each filter comes with its own template (input, select, autocomplete etc)
- but we wanted to keep using them as simple as possible

```
list_filter = [  
    'barcode', 'status', 'imei', 'sn', 'model', 'purchase_order',  
    'model__category', 'loan_end_date', 'niw', 'model__manufacturer',  
]
```

Back Office Assets

Actions

Go 0 of 100 selected

Filters

Barcode

Status

All

Imei

SN

Model

Purchase order

Hostname

Required support

All

Region

Warehouse

<input type="checkbox"/>	Status	Barcode	Purchase order	Model
<input type="checkbox"/>	free	bo100000227		[Edit]
<input type="checkbox"/>	new	bo100000226		[Edit]
<input type="checkbox"/>	new	bo100000225	(None)	[Edit]
<input type="checkbox"/>	new	bo100000224	(None)	[New]
<input type="checkbox"/>	new	bo100000222	(None)	[Edit]
<input type="checkbox"/>	new	bo100000221	(None)	[Edit]
<input type="checkbox"/>	new	bo100000220	(None)	[Edit]
<input type="checkbox"/>	new	bo100000219	(None)	[Edit]
<input type="checkbox"/>	new	bo100000218	(None)	[Edit]
<input type="checkbox"/>	new	bo100000217		[Edit]
<input type="checkbox"/>	new	bo100000216	(None)	[Edit]
<input type="checkbox"/>	new	bo100000215	(None)	[Edit]
<input type="checkbox"/>	new	bo100000214	(None)	[Edit]
<input type="checkbox"/>	new	bo100000213	(None)	[Edit]

Data center assets

Actions

Go 0 of 100 selected

Filters

Hostname

Service env

Configuration class path

Configuration class module

MAC Address

IP

Vulnerability patch deadline

Start YYYY-MM End YYYY-MM-

Security scan vulnerabilities

Security scan is patched

All

Invoice number

Invoicing date

<input type="checkbox"/>	Hostname	Status	Barcode	Model
<input type="checkbox"/>	ralph421.allegro.pl	in use	(None)	[Edit]
<input type="checkbox"/>	ralph421.allegro.pl	new	dc100000421	[Edit]
<input type="checkbox"/>	ralph420.allegro.pl	new	dc100000420	[Edit]
<input type="checkbox"/>	ralph419.allegro.pl	in use	dc100000419	[Edit]
<input type="checkbox"/>	ralph418.allegro.pl	new	dc100000418	[Edit]
<input type="checkbox"/>	ralph417.allegro.pl	new	dc100000417	[Edit]
<input type="checkbox"/>	ralph416.allegro.pl	new	dc100000416	[Edit]
<input type="checkbox"/>	ralph415.allegro.pl	new	dc100000415	[Edit]
<input type="checkbox"/>	ralph414.allegro.pl	new	dc100000414	[Edit]
<input type="checkbox"/>	ralph413.allegro.pl	new	dc100000413	[Edit]
<input type="checkbox"/>	ralph412.allegro.pl	new	dc100000412	[Edit]
<input type="checkbox"/>	ralph411.allegro.pl	new	dc100000411	[Edit]
<input type="checkbox"/>	ralph410.allegro.pl	new	dc100000410	[Edit]
<input type="checkbox"/>	ralph409.allegro.pl	new	dc100000409	[Edit]

Invoice number i

Use ; or | separators to search for multiple values

Invoice date

Filters

Hostname

 i

Service env

 i

Configuration class path i

Configuration class module

 i

MAC Address

Service env

Ralph - prod i



Ralph - dev i



Ralph - test i



How to register our own filter classes?

- `FieldListFilter` provides a way to register our own filter classes (used if some condition pass)

```
from django.contrib.admin.filters import FieldListFilter

field_filter_mapper = [
    (lambda f: bool(f.choices), ChoicesListFilter),
    (lambda f: isinstance(f, models.IntegerField), NumberListFilter),
    (lambda f: isinstance(f, (models.CharField, models.TextField)), TextListFilter),
    (lambda f: isinstance(f, (models.ForeignKey, models.ManyToManyField)),
     RelatedAutocompleteFieldListFilter),
]

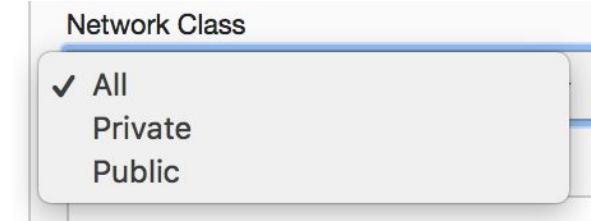
for func, filter_class in field_filter_mapper:
    FieldListFilter.register(func, filter_class, take_priority=True)
```

Using dedicated filters for field

```
class NetworkClassFilter(ChoicesListFilter):
    _choices_list = [('private', _('Private')), ('public', _('Public'))]

    def queryset(self, request, queryset):
        if not self.value():
            return queryset
        if self.value().lower() == 'private':
            queryset = queryset.filter(PRIVATE_NETWORK_FILTER)
        elif self.value().lower() == 'public':
            queryset = queryset.exclude(PRIVATE_NETWORK_FILTER)
        return queryset

class IPAddressAdmin(RalphAdmin):
    list_filter = [..., ('address', NetworkClassFilter), ...]
```



API

API

- based on [Django Rest Framework](#)
- goal: (almost) every model and action visible in UI might be done in API
- respect permissions (per field)
- keep it as declarative as possible

API - Django Rest Framework basics

- Serializers
- Viewsets
- Routes

```
from rest_framework import routers, serializers, viewsets

class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ('url', 'username', 'email', 'groups')

class UserViewSet(viewsets.ModelViewSet):
    queryset = User.objects.all().order_by('-date_joined')
    serializer_class = UserSerializer

router = routers.DefaultRouter()
router.register(r'users', UserViewSet)

urlpatterns = [url(r'^', include(router.urls))]
```

API - serializers

- handles permissions (per fields and related objects)
- request and user object easily accessible in each related serializer
- nested and choice fields tweaks
- CQRS - RalphAPISaveSerializer

```
class RalphAPISerializer(  
    TaggitSerializer,  
    RelatedObjectsPermissionsSerializerMixin,  
    PermissionsPerFieldSerializerMixin,  
    serializers.HyperlinkedModelSerializer,  
    metaclass=RalphAPISerializerMetaclass  
):  
    pass
```

API - viewsets

- handles `select_related` and `prefetch_related` in declarative way
- lot of filter backends out-of-the-box (filters from admin, tags, custom fields, `__` lookup)
- handles `save_serializer_class` or create it dynamically if not provided

```
class RalphAPIViewSet(  
    QuerysetRelatedMixin,  
    AdminSearchFieldsMixin,  
    viewsets.ModelViewSet,  
    metaclass=RalphAPIViewSetMetaclass  
):  
    pass
```

```
class IPAddressSerializer(RalphAPISerializer):
    network = NetworkSimpleSerializer()

    class Meta:
        model = IPAddress
        depth = 1

class IPAddressSaveSerializer(RalphAPISaveSerializer):
    class Meta:
        model = IPAddress

    def validate_dhcp_expose(self, value):
        # validation ...
        return value

class IPAddressViewSet(RalphAPIViewSet):
    queryset = IPAddress.objects.all()
    serializer_class = IPAddressSerializer
    save_serializer_class = IPAddressSaveSerializer
    select_related = ['network']
    prefetch_related = ['ethernet', 'ethernet__base_object']
    filter_fields = ['hostname', 'network__address', 'ethernet__mac']
```

```
class VirtualServerFilterSet(django_filters.FilterSet):
    configuration_path = django_filters.CharFilter(
        name='configuration_path__path'
    )
    ip = django_filters.CharFilter(
        name='ethernet_set__ipaddress__address'
    )

    class Meta:
        model = VirtualServer
        fields = ['configuration_path__module__name']

class VirtualServerViewSet(RalphAPIViewSet):
    # ...
    extended_filter_fields = {
        'service': ['service_env__service__uid', 'service_env__service__name'],
        'env': ['service_env__environment__name'],
    }
    additional_filter_class = VirtualServerFilterSet
```

Bulk edit

Bulk edit

- allow to edit multiple items at once
- respect permissions (per field)
- custom Django's Admin action: `bulk_edit_action`

Back Office Assets

Actions

✓ -----

Delete selected Back Office Assets

Invoice report

Bulk edit

Release asset transition

Loan asset transition

Return asset transition

Status

Back Office Assets

ID	Status	Barcode	Imei	Hostname
32	new	bo100000031	274675226767018	c0031
20	in use	bo100000019	513468310185078	c0019
14	installed	bo100000013	651418427773433	c0013

Back Office Assets

Back to list

+ Add Back Office Asset

Other actions ▾

ID	Status	Barcode	Imei	Hostname	Model	User	Owner
32	in service	bo100000031	274675226767018	c0031	[Mobile Phone] Apple ...	Lori Brown (eric...)	Laura Gates (c)
20	loan	bo100000019	513468310185078	c0019	[Mobile Phone] Apple ...	Nicole Morales ...	Jeremy Vasqu
14	in use	bo100000013	651418427773433	c0013	[Mobile Phone] Apple ...	Terry Singh (aar...)	Michael Wagn

Bulk edit - how?

```
@register(BackOfficeAsset)
class BackOfficeAssetAdmin(BulkEditChangeListMixin, RalphAdmin):
    # ...
    bulk_edit_list = [
        'licences', 'status', 'barcode', 'imei', 'hostname', 'model',
        'purchase_order', 'user', 'owner', 'warehouse', 'sn', 'region',
        'property_of', 'remarks',
    ]
    bulk_edit_no_fillable = ['barcode', 'sn', 'imei', 'hostname']
    # ...
```

Bulk edit - how?

- using Django's admin `list_editable` property underneath
- override `get_queryset` to narrow it down only to selected objects
- override `get_list_display` to display editable fields (with respect to permissions)

Conclusions

Conclusions

- Lot of work with metaclasses and/or content types to make something generic
- It's not so easy to customize/extend Django's admin!
- Do we regret it? NO!

Q&A

<https://github.com/allegro/ralph>

Slides: https://mkurek.com/django_admin.pdf

Mateusz Kurek



mkurek