

Restful API Development with Django Rest Framework and Celery

Kenapa harus Django?

Fitur - Fitur Django

- Generated Admin
- ORM
- Template Engine
- Console
- ModelForm & Validation
- Built-in Authentication & Authorization
- Powerful Migration Tool
- Middleware
- Security
- Static Management
- A lot of battery included
- A lot of third party package
- Et cetera

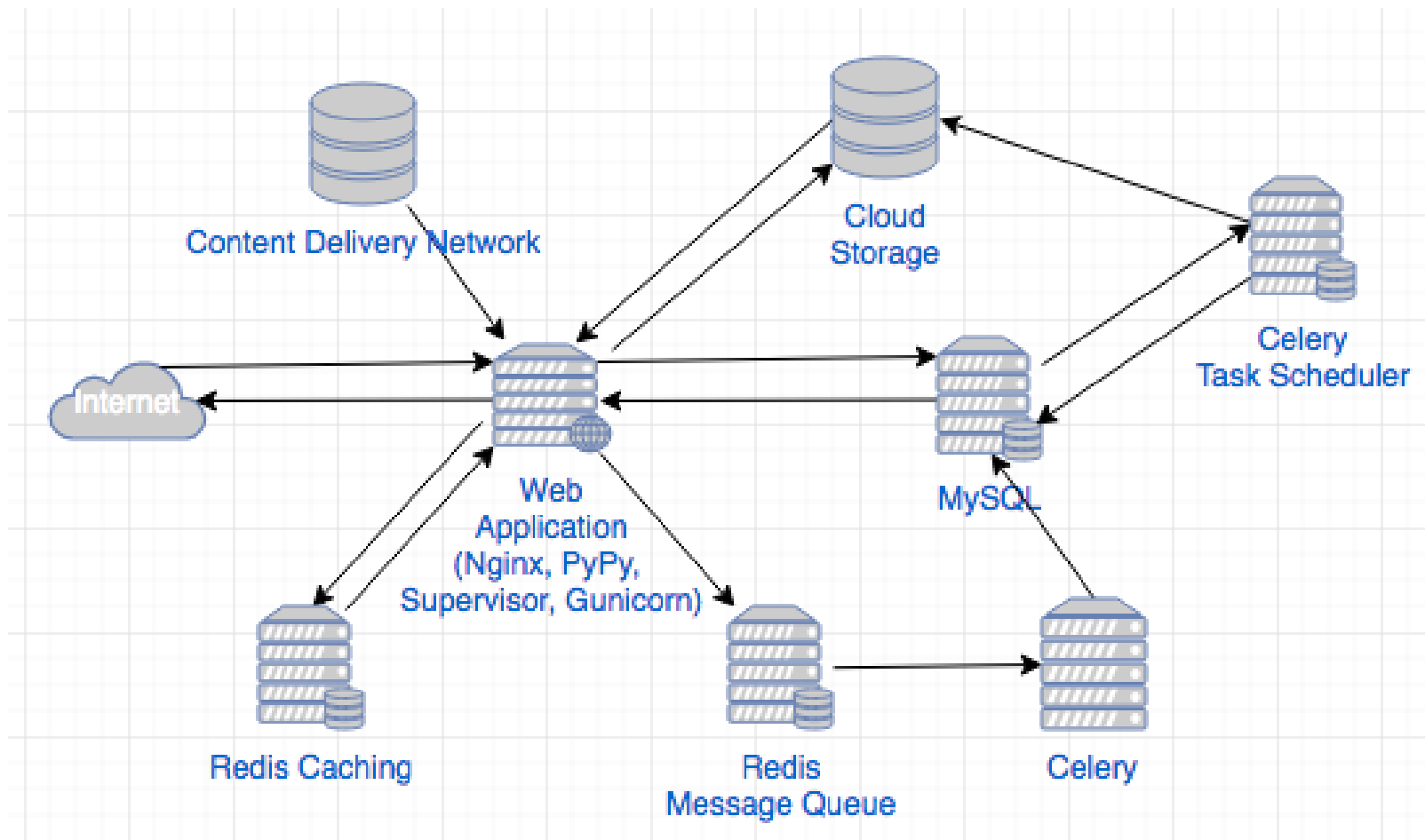
Fitur - Fitur Django Rest Framework

- Generated Api Console
- Serializer
- Request & Response
- Pagination
- Built-in Authentication, Permission and Authorization
- Throttling and Filtering
- Testing
- A lot of battery included
- A lot of third party package
- Et cetera

Fitur - Fitur Celery

- Various Broker (Redis, SQS, RabbitMQ, etc.)
- Periodic Task
- Task Queue
- Easy Integration with Populer Framework (Django, Flask, Falcon, Bottle)
- Monitoring with Flower
- Et cetera.

Contoh Arsitektur Django



Saatnya Bedah Kode !

Setting Django

```
25
26 # SECURITY WARNING: don't run with debug turned on in production!
27 DEBUG = True
28
29 ALLOWED_HOSTS = []
30
31
32 # Application definition
33
34 INSTALLED_APPS = [
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41     'rest_framework',
42     'rest_framework.authtoken',
43     'blog'
44 ]
45
46 MIDDLEWARE = [
47 ]
48
49 ROOT_URLCONF = 'myapp.urls'
50
51 TEMPLATES = [
52 ]
53
54 WSGI_APPLICATION = 'myapp.wsgi.application'
55
56
57 # Database
58 # https://docs.djangoproject.com/en/1.10/ref/settings/#databases
59
60 DATABASES = {
61     'default': {
62         'ENGINE': 'django.db.backends.mysql',
63         'NAME': 'myapp',
64         'USER': 'root',
65         'PASSWORD': '',
66     }
67 }
68
69
70 # Password validation
71 # https://docs.djangoproject.com/en/1.10/ref/settings/#auth-password-validators
72
73 AUTH_PASSWORD_VALIDATORS = [
74     {
75         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
76     },
77     {
78         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
79     },
80 ]
```


Setting Django Rest Framework

```
128 # Rest Framework
129
130 REST_FRAMEWORK = {
131     'DEFAULT_PERMISSION_CLASSES': [
132         'rest_framework.permissions.IsAuthenticatedOrReadOnly',
133         'rest_framework.permissions.DjangoModelPermissionsOrAnonReadOnly'
134     ],
135     'DEFAULT_AUTHENTICATION_CLASSES': (
136         'rest_framework.authentication.TokenAuthentication',
137         'rest_framework.authentication.SessionAuthentication',
138     ),
139     'PAGE_SIZE': 10
140 }
141
```

Setting Celery

```
142 # Celery Setting
143
144 BROKER_URL = 'redis://localhost:6379/0'
145 CELERY_ACCEPT_CONTENT = ['pickle']
146 CELERY_TASK_SERIALIZER = 'pickle'
147 CELERY_TIMEZONE = 'Asia/Jakarta'
148
149 CELERYBEAT_SCHEDULE = {
150     'send-random-article-every-3-seconds': {
151         'task': 'blog.tasks.send_random_article',
152         'schedule': timedelta(seconds=3),
153     },
154 }
155
```

Model

```
1 from __future__ import unicode_literals
2
3 from django.db import models
4 from django.contrib.auth.models import *
5
6 # Create your models here.
7 class Category (models.Model):
8     name = models.CharField(max_length=50)
9
10     def __unicode__(self):
11         return self.name
12
13 class Posts (models.Model):
14     title = models.CharField(max_length=100)
15     body = models.TextField()
16     author = models.ForeignKey(User)
17     category = models.ManyToManyField(Category)
18     created_at = models.DateTimeField(auto_now_add=True)
19
20     def __unicode__(self):
21
22         return self.title + " - " + self.author.username
23
24     def category_list(self):
25         return ', '.join([a.name for a in self.category.all()])
26
27 class Comment (models.Model):
28     post = models.ForeignKey(Posts)
29     user = models.ForeignKey(User)
30     body = models.TextField()
31     created_at = models.DateTimeField(auto_now_add=True)
32
33
34     def __unicode__(self):
35
36         return self.user.username + " - " + self.body + " on " + self.post.body
37
38
```

Migration

```
1  # -*- coding: utf-8 -*-
2  # Generated by Django 1.10.5 on 2017-04-26 15:57
3  from __future__ import unicode_literals
4
5  from django.conf import settings
6  from django.db import migrations, models
7  import django.db.models.deletion
8
9
10 class Migration(migrations.Migration):
11
12     initial = True
13
14     dependencies = [
15         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
16     ]
17
18     operations = [
19         migrations.CreateModel(
20             name='Category',
21             fields=[
22                 ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
23                 ('name', models.CharField(max_length=50)),
24             ],
25         ),
26         migrations.CreateModel(
27             name='Comment',
28             fields=[
29                 ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
30                 ('body', models.TextField()),
31                 ('created_at', models.DateTimeField(auto_now_add=True)),
32             ],
33         ),
34         migrations.CreateModel(
35             name='Posts',
36             fields=[
37                 ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
38                 ('title', models.CharField(max_length=100)),
39                 ('body', models.TextField()),
40                 ('created_at', models.DateTimeField(auto_now_add=True)),
41                 ('author', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
42                 ('category', models.ManyToManyField(to='blog.Category')),
43             ],
44         ),
45         migrations.AddField(
46             model_name='comment',
47             name='post',
48             field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='blog.Posts'),
49         ),
50         migrations.AddField(
51             model_name='comment',
52             name='user',
53             field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL),
```

Admin

```
1 from django.contrib import admin
2 from .models import *
3
4 # Register your models here.
5
6 admin.site.register(Category)
7
8 class PostAdmin(admin.ModelAdmin):
9     list_display = ('title', 'body', 'category_list', 'author', 'created_at')
10    search_fields = ['title', 'body']
11    date_hierarchy = 'created_at'
12
13 admin.site.register(Posts, PostAdmin)
14
15 class CommentAdmin(admin.ModelAdmin):
16     list_display = ('post', 'user', 'body', 'created_at')
17     search_fields = ['body', 'post']
18     date_hierarchy = 'created_at'
19
20 admin.site.register(Comment, CommentAdmin)
```

Signal

```
1 from django.dispatch import receiver
2 from django.db.models.signals import post_save
3 from .models import Posts
4 from .tasks import convert_to_pdf
5
6 import json
7
8 @receiver(post_save, sender=Posts)
9 def call_pdf_converter(sender, instance, *args, **kwargs):
10     if kwargs.get("created"):
11         print "call_pdf_converter..."
12         convert_to_pdf.delay(instance)
13
14     return True
15
```

Celery Task

```
4 from django.core.mail import send_mail
5
6 import time
7 from reportlab.lib.enums import TA_JUSTIFY
8 from reportlab.lib.pagesizes import letter
9 from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
10 from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
11 from reportlab.lib.units import inch
12
13 @shared_task
14 def convert_to_pdf(instance):
15     print "calling convert_to_pdf()..."
16
17     print instance.id
18     print instance.title
19
20     # disini nanti ada kode buat convert ke pdf dari artikel yang diubah atau ditambah
21
22     doc = SimpleDocTemplate("assets/pdf/"+instance.title+"_"+str(instance.id)+".pdf", pagesize=letter,
23                             rightMargin=72, leftMargin=72,
24                             topMargin=72, bottomMargin=18)
25
26     Story=[]
27     logo = "python_logo.png"
28     magName = "Pythonista"
29     issueNum = 12
30     subPrice = "99.00"
31     limitedDate = "03/05/2010"
32     freeGift = "tin foil hat"
33     formatted_time = time.ctime()
34     full_name = "Mike Driscoll"
35     address_parts = ["411 State St.", "Marshalltown, IA 50158"]
36
37     styles=getSampleStyleSheet()
38     styles.add(ParagraphStyle(name='Justify', alignment=TA_JUSTIFY))
39     ptext = '<font size=20>%s</font>' % instance.title
40     Story.append(Paragraph(ptext, styles["Normal"]))
41     Story.append(Spacer(1, 12))
42
43     ptext = '<font size=10>created by %s at %s</font>' % (instance.author.username, instance.created_at)
44
45     Story.append(Paragraph(ptext, styles["Normal"]))
46     Story.append(Spacer(1, 24))
47
48     ptext = instance.body
49     Story.append(Paragraph(ptext, styles["Justify"]))
50     Story.append(Spacer(1, 12))
51
52     doc.build(Story)
53
54     return True
55
```

Serializer

```
1 from django.contrib.auth.models import User
2 from rest_framework import serializers
3 from .models import Category, Posts, Comment
4
5
6 class UserSerializer(serializers.ModelSerializer):
7
8     class Meta:
9         model = User
10        fields = ('url', 'email', 'first_name', 'last_name', 'is_superuser')
11        read_only_fields = ("is_superuser",)
12
13 class CategorySerializer(serializers.HyperlinkedModelSerializer):
14     class Meta:
15         model = Category
16         fields = ['url', 'id', 'name']
17
18 class PostSerializer(serializers.HyperlinkedModelSerializer):
19
20     class Meta:
21         model = Posts
22
23         fields = ["url", "id", 'title', 'body', 'author', 'category', 'created_at']
24         read_only_fields = ('author',)
25
26 class CommentSerializer(serializers.HyperlinkedModelSerializer):
27     class Meta:
28         model = Comment
29         fields = ['url', 'id', 'post', 'body', 'user', 'created_at']
30         read_only_fields = ("user",)
```


Permission

```
1 from rest_framework import permissions
2
3 class UserPermission (permissions.BasePermission):
4
5     def has_permission(self, request, view):
6         if view.action in ['list', 'retrieve']:
7             return True
8         elif view.action in ['update', 'partial_update']:
9             return request.user.is_authenticated() or request.user.is_staff
10        elif view.action in ['create', 'destroy']:
11            return request.user.is_staff
12        else:
13            return False
14
15    def has_object_permission(self, request, view, obj):
16        if view.action in ['retrieve']:
17            return True
18        elif view.action in ['update', 'partial_update']:
19            return (obj == request.user or request.user.is_staff)
20        elif view.action in ['destroy']:
21            return request.user.is_staff
22        else:
23            return False
24
25 class PostsPermission (permissions.BasePermission):
26
27     def has_permission(self, request, view):
28         if view.action in ['list', 'retrieve']:
29             return True
30        elif view.action in ['create', 'destroy', 'update', 'partial_update']:
31            return request.user.is_authenticated() or request.user.is_staff
32        else:
33            return False
34
35    def has_object_permission(self, request, view, obj):
36        if view.action in ['retrieve']:
37            return True
38        elif view.action in ['update', 'partial_update', 'destroy']:
39            return (obj.author == request.user or request.user.is_staff)
40        else:
41            return False
42
43 class CommentPermission (permissions.BasePermission):
44
45     def has_permission(self, request, view):
46         if view.action in ['list', 'retrieve']:
47             return True
48        elif view.action in ['create', 'destroy', 'update', 'partial_update']:
49            return request.user.is_authenticated() or request.user.is_staff
50        else:
51            return False
52
53    def has object permission(self, request, view, obj):
```

ViewSet

```
1 from django.shortcuts import render
2 from django.contrib.auth.models import User
3 from rest_framework import viewsets
4 from .serializers import CategorySerializer, PostSerializer, CommentSerializer, UserSerializer
5 from .models import Category, Posts, Comment
6 from .permissions import UserPermission, PostsPermission, CommentPermission
7 from rest_framework.response import Response
8 from rest_framework import status
9
10 import json
11
12 # Create your views here.
13
14 class UserViewSet(viewsets.ModelViewSet):
15     queryset = User.objects.all().order_by('-id')
16     serializer_class = UserSerializer
17     permission_classes = (UserPermission,)
18
19 class CategoryViewSet(viewsets.ModelViewSet):
20     queryset = Category.objects.all()
21     serializer_class = CategorySerializer
22
23 class PostViewSet(viewsets.ModelViewSet):
24     queryset = Posts.objects.all()
25     serializer_class = PostSerializer
26     permission_classes = (PostsPermission,)
27
28     def create(self, request):
29         serializer = self.get_serializer(data=request.data)
30
31         if serializer.is_valid():
32             serializer.save(author=self.request.user)
33
34             return Response(serializer.data, status.HTTP_201_CREATED)
35
36         return Response(dict(serializer.errors), status.HTTP_400_BAD_REQUEST)
37
38
39 class CommentViewSet(viewsets.ModelViewSet):
40     queryset = Comment.objects.all()
41     serializer_class = CommentSerializer
42     permission_classes = (CommentPermission,)
43
44     def create(self, request):
45         serializer = self.get_serializer(data=request.data)
46
47         if serializer.is_valid():
48             serializer.save(user=self.request.user)
49
50             return Response(serializer.data, status.HTTP_201_CREATED)
51
52         return Response(dict(serializer.errors), status.HTTP_400_BAD_REQUEST)
53
```

Url Config

```
1 from django.conf.urls import include, url
2 from django.contrib import admin
3 from rest_framework import routers
4 from rest_framework.auth_token import views
5 from blog import views as blog_views
6
7 router = routers.DefaultRouter()
8 router.register(r'users', blog_views.UserViewSet)
9 router.register(r'categories', blog_views.CategoryViewSet)
10 router.register(r'posts', blog_views.PostViewSet)
11 router.register(r'comments', blog_views.CommentViewSet)
12
13 urlpatterns = [
14     url(r'^admin/', admin.site.urls),
15     url(r'^', include(router.urls)),
16     url(r'^api-auth/', include('rest_framework.urls',
17                               namespace='rest_framework')),
18     url(r'^api-token-auth/', views.obtain_auth_token)
19 ]
20
```

Contoh Halaman Admin

← → ↺

localhost:8000/admin/

🔑 ☆ 🖨️ ⌂ 🏠 📄

Django administration

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

Site administration

AUTH TOKEN

Tokens

+ Add 🖋️ Change

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add 🖋️ Change

Users

+ Add 🖋️ Change

BLOG

Categories

+ Add 🖋️ Change

Comments

+ Add 🖋️ Change

Postss

+ Add 🖋️ Change

Recent actions

My actions

+ ridwanbejo - asdfasdfsdfasdfsdf on Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a. Donec sollicitudin molestie malesuada. Curabitur aliquet quam id dui posuere blandit. Curabitur al

Comment

+ kresnaguluh - asdfasdfsdfasdf on Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a. Donec sollicitudin molestie malesuada. Curabitur aliquet quam id dui posuere blandit. Curabitur aliquet q

Comment

+ kresnaguluh - asdfasdfsdf on Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a. Donec sollicitudin molestie malesuada. Curabitur aliquet quam id dui posuere blandit. Curabitur aliquet quam

Comment

+ Lorem ipsum sit dolor amet - ridwanbejo

Posts

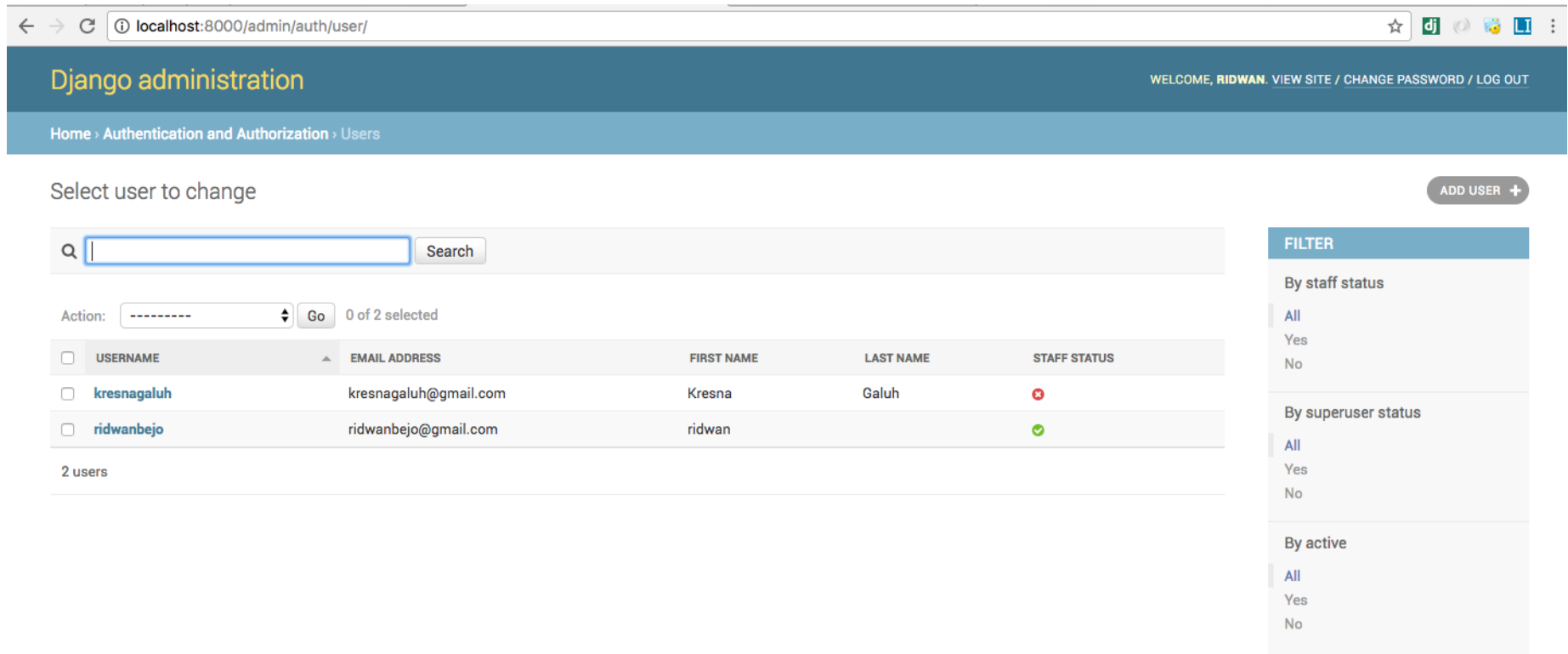
+ test - kresnaguluh

Posts

✖ a - ridwanbejo

Posts

Contoh Halaman Admin



The screenshot displays the Django administration interface for user management. The browser address bar shows `localhost:8000/admin/auth/user/`. The page header includes the Django logo and navigation links: `WELCOME, RIDWAN. VIEW SITE / CHANGE PASSWORD / LOG OUT`. The breadcrumb trail indicates the current location: `Home > Authentication and Authorization > Users`.

Below the header, there is a section titled "Select user to change" with an "ADD USER +" button. A search bar with a magnifying glass icon and a "Search" button is provided. Below the search bar, an "Action:" dropdown menu is set to "-----", followed by a "Go" button and the text "0 of 2 selected".

The main content area features a table with the following columns: `USERNAME`, `EMAIL ADDRESS`, `FIRST NAME`, `LAST NAME`, and `STAFF STATUS`. The table lists two users:

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	kresnagaluh	kresnagaluh@gmail.com	Kresna	Galuh	✖
<input type="checkbox"/>	ridwanbejo	ridwanbejo@gmail.com	ridwan		✔

Below the table, it indicates "2 users".

On the right side, there is a "FILTER" sidebar with three sections:

- By staff status**: All, Yes, No
- By superuser status**: All, Yes, No
- By active**: All, Yes, No

Contoh Halaman Admin

← → ↺ ⓘ localhost:8000/admin/blog/posts/ ☆ dj 🌐 🏠 LI ⋮

Django administration

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

Home » Blog » Postss

Select posts to change

ADD POSTS +

Q |

Search

< 2017

Action:

Go

 0 of 14 selected

<input type="checkbox"/>	TITLE	BODY	CATEGORY LIST	AUTHOR	CREATED AT
<input type="checkbox"/>	test-1	lorem ipsum sit dolor amet	Kuliner, Massage	kresnagalah	April 29, 2017, 12:39 a.m.
<input type="checkbox"/>	a	a	Kuliner, Massage	ridwanbejo	April 29, 2017, 12:38 a.m.
<input type="checkbox"/>	Lorem Ipsum	Nulla quis lorem ut libero malesuada feugiat. Vivamus suscipit tortor eget felis porttitor volutpat. Curabitur arcu erat, accumsan id imperdiet et, porttitor at sem. Donec rutrum congue leo eget malesuada. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec velit neque, auctor sit amet aliquam vel, ullamcorper sit amet ligula. Sed porttitor lectus nibh. Sed porttitor lectus nibh. Praesent sapien massa, convallis a pellentesque nec, egestas non nisi. Cras ultricies ligula sed magna dictum porta. Curabitur aliquet quam id dui posuere blandit. Proin eget tortor risus. Sed porttitor lectus nibh. Nulla quis lorem ut libero malesuada feugiat. Nulla quis lorem ut libero malesuada feugiat. Sed porttitor lectus nibh. Proin eget tortor risus. Donec sollicitudin molestie malesuada. Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quis lorem ut libero malesuada feugiat. Vestibulum ac diam sit amet quam vehicula elementum sed sit amet dui. Nulla porttitor accumsan tincidunt. Nulla quis lorem ut libero malesuada feugiat. Vivamus magna justo, lacinia eget consectetur sed, convallis at tellus. Pellentesque in ipsum id orci porta dapibus. Nulla porttitor accumsan tincidunt. Proin eget tortor risus. Vestibulum ac diam sit amet quam vehicula elementum sed sit amet dui. Quisque velit nisi, pretium ut lacinia in, elementum id enim. Quisque velit nisi, pretium ut lacinia in, elementum id enim. Nulla porttitor accumsan tincidunt. Vestibulum ac diam sit amet quam vehicula elementum sed sit amet dui. Nulla quis lorem ut libero malesuada feugiat. Curabitur non nulla sit amet nisl tempus convallis quis ac lectus. Vivamus magna justo, lacinia eget consectetur sed, convallis at tellus. Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a. Nulla porttitor accumsan tincidunt. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur aliquet quam id dui posuere blandit. Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a. Vivamus suscipit tortor eget felis porttitor volutpat. Nulla quis lorem ut libero malesuada feugiat. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus magna justo, lacinia eget consectetur sed, convallis at tellus. Nulla quis lorem ut libero malesuada feugiat. Pellentesque in ipsum id orci porta dapibus. Cras ultricies ligula sed magna dictum porta. Donec rutrum congue leo eget malesuada. Vivamus suscipit tortor eget felis porttitor volutpat. Quisque velit nisi, pretium ut lacinia in, elementum id enim.	Kuliner, Massage	ridwanbejo	April 29, 2017, 12:36 a.m.
<input type="checkbox"/>	bejo	bejo	Massage	ridwanbejo	April 29, 2017, 12:34 a.m.

 Codepolitan

Codepolitan Meetup, 2 Mei 2016

Contoh Halaman Admin

← → ↺

localhost:8000/admin/blog/posts/57/change/

☆ dj ↻ 🌐 LT ⋮

Django administration

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

Home » Blog » Posts » Lorem Ipsum - ridwanbejo

Change posts

HISTORY

Title:

Lorem Ipsum

Body:

Nulla quis lorem ut libero malesuada feugiat. Vivamus suscipit tortor eget felis porttitor volutpat. Curabitur arcu erat, accumsan id imperdiet et, porttitor at sem. Donec rutrum congue leo eget malesuada. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec velit neque, auctor sit amet aliquam vel, ullamcorper sit amet ligula. Sed porttitor lectus nibh. Sed porttitor lectus nibh. Praesent sapien massa, convallis a pellentesque nec, egestas non nisi. Cras ultricies ligula sed magna dictum porta. Curabitur aliquet quam id dui posuere blandit.

Proin eget tortor risus. Sed porttitor lectus nibh. Nulla quis lorem ut libero malesuada feugiat. Nulla quis lorem ut libero malesuada feugiat. Sed porttitor lectus nibh. Proin eget tortor risus. Donec sollicitudin molestie malesuada. Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a. Lorem ipsum dolor sit

Author:

ridwanbejo

Category:

Kuliner

Massage

+

Hold down "Control", or "Command" on a Mac, to select more than one.

Delete

Save and add another

Save and continue editing

SAVE

Contoh Halaman Konsol API

Django REST framework

kresnagaluh ▾

API Root / User / User

User Instance

GET ▾

GET /users/2/

HTTP 200 OK

Allow: GET, PUT, PATCH, DELETE, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "url": "http://localhost:8000/users/2/",
  "email": "kresnagaluh@gmail.com",
  "first_name": "Kresna",
  "last_name": "Galuh",
  "is_superuser": false
}
```

Raw data

HTML form

Email address

kresnagaluh@gmail.com

First name

Kresna

Last name

Galuh

PUT

Contoh Halaman Konsol API

Django REST framework kresnagaluh ▾

Api Root / Post

Post List GET ▾

« 1 2 »

GET /posts/

HTTP 200 OK
Allow: GET, POST, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "count": 14,
  "next": "http://localhost:8000/posts/?page=2",
  "previous": null,
  "results": [
    {
      "url": "http://localhost:8000/posts/43/",
      "id": 43,
      "title": "test asda sdas das das",
      "body": "test adsa sdasd asd a",
      "author": "http://localhost:8000/users/1/",
      "category": [
        "http://localhost:8000/categories/2/"
      ],
      "created_at": "2017-04-27T15:05:18.284426Z"
    },
    {
      "url": "http://localhost:8000/posts/44/",
      "id": 44,
      "title": "Lorem ipsum sit dolor amet",
      "body": "Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a. Donec sollicitudin molestie malesuada. Curabitur aliquet quam id dui posu",
      "author": "http://localhost:8000/users/1/"
    }
  ]
}
```

Contoh Halaman Konsol API

Django REST framework

kresnagaluh ▾

Post Instance

DELETE

GET ▾

GET /posts/49/

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "url": "http://localhost:8000/posts/49/",
  "id": 49,
  "title": "asd",
  "body": "asd",
  "author": "http://localhost:8000/users/2/",
  "category": [],
  "created_at": "2017-04-28T11:09:54.745952Z"
}
```

Raw data

HTML form

Title

asd

Body

asd

Category

Kuliner
Massage

Deployment

- VPS: Linode, Digital Ocean, Azure VM, AWS EC2, etc.
- PaaS: Heroku, Python Anywhere, OpenShift, IBM BlueMix
- Standard Stack: Nginx, Gunicorn, Supervisor
- Et cetera.

DEMO !

Tanya Jawab ;()

Penutup

Source code dan slide: <https://github.com/ridwanbejo/codepolitan-meetup-april-2017>

Kontak:

- LinkedIn: <https://www.linkedin.com/in/ridwan-fadjar-79781756/>
- Github: <https://www.github.com/ridwanbejo>
- Email: ridwanbejo@gmail.com
- Codepolitan: <https://www.codepolitan.com/coder/ridwan>