The Entrepreneurship Network

Date:- 27-01-2022

Document Name:- Task-2 – to build Rest API's for a website using Django

Name:- Arshad Bagde

Employee Id:- TEN/PD/1855

Batch No:- Dec-2021

Email Id:- abagde61@gmail.com

College Name:- National Institute of Information Technology, Chennai.

Question:-

Greetings!!
Here's your 2nd Core task.

- • As the next steps, you are required to build Rest API's for a website using django.
- • To minimize any back and forth discussions with multiple teams and boost productivity in the short span of development time, the intention is to reproduce the backend model of a website that is already used by our generation frequently. Example of one such website is Amazon.
- • As part of the technical task, the assignment for you is to create Back-end with multiple API's that could be used in developing a website like TEN (https://www.entrepreneurshipnetwork.net/)
- • As a developer, you are required to think out of the box, plan your development strategy and try developing API's in a way that they support maximum number of features being used in the website.
- • Focus should be on maximizing results from each API, reduce any redundancy and use Best

Practices.

- • You are also required to make sure that the API's can be tested through Swagger.
- • Following a proper code structure and handling exceptions should be prioritized and would attract more points.
- • Please note that Front-end team would not be involved in the task assigned to you.
- • Extra Points would be awarded for using the micro-services architecture..
- • The development time is 15 – 20 days.
- • You will be aligned with the Technical Lead to understand your problem solving capabilities; code quality review, software testing etc. and the points for the same would be awarded.

Following are few points to keep in mind that can fetch you extra points:
➢ Adhere to WordPress Coding Standards
➢ It is best to choose a plugin or theme that is updated frequently and has high satisfaction ratings.
➢ Practices like Readability, Reliability, and Flexibility.
➢ While pushing the code to your respective repository, plan to include a proper documentation

about your work.
If you have any questions, please feel free to reach out to us and we'll be more than happy to help you. We are looking forward to working with you and seeing you achieve great things!
Regards,
Team TEN

Answer:-

~~~
# [Django REST framework][docs]

# Installation

Install using `pip`...

    pip install djangorestframework

Add `'rest_framework'` to your `INSTALLED_APPS` setting.

    INSTALLED_APPS = [
        ...
        'rest_framework',
    ]

# Example

Let's take a look at a quick example of using REST framework to build a simple
model-backed API for accessing users and groups.

Startup up a new project like so...

    pip install django
    pip install djangorestframework
    django-admin startproject example .
    ./manage.py migrate
    ./manage.py createsuperuser


Now edit the `example/urls.py` module in your project:

```python
from django.urls import path, include
from django.contrib.auth.models import User
from rest_framework import serializers, viewsets, routers

# Serializers define the API representation.
class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ['url', 'username', 'email', 'is_staff']
~~~

~~~~
# ViewSets define the view behavior.
class UserViewSet(viewsets.ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer


# Routers provide a way of automatically determining the URL conf.
router = routers.DefaultRouter()
router.register(r'users', UserViewSet)


# Wire up our API using automatic URL routing.
# Additionally, we include login URLs for the browsable API.
urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls',
namespace='rest_framework')),
]
```

We'd also like to configure a couple of settings for our API.

Add the following to your `settings.py` module:

```python
INSTALLED_APPS = [
    ...  # Make sure to include the default installed apps here.
    'rest_framework',
]

REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.DjangoModelPermissionsOrAnonReadOnly',
    ]
}
```

That's it, we're done!

    ./manage.py runserver

You can now open the API in your browser at `http://127.0.0.1:8000/`, and view
your new 'users' API. If you use the `Login` control in the top right corner
you'll also be able to add, create and delete users from the system.

You can also interact with the API using command line tools such as
[`curl`](https://curl.haxx.se/). For example, to list the users endpoint:
~~~~

```
    $ curl -H 'Accept: application/json; indent=4' -u admin:password
http://127.0.0.1:8000/users/
    [
        {
            "url": "http://127.0.0.1:8000/users/1/",
            "username": "admin",
            "email": "admin@example.com",
            "is_staff": true,
        }
    ]

Or to create a new user:

    $ curl -X POST -d username=new -d email=new@example.com -d is_staff=false
-H 'Accept: application/json; indent=4' -u admin:password
http://127.0.0.1:8000/users/
    {
        "url": "http://127.0.0.1:8000/users/2/",
        "username": "new",
        "email": "new@example.com",
        "is_staff": false,
    }
```

```python
#! /usr/bin/env python3
import sys

import pytest


def split_class_and_function(string):
    class_string, function_string = string.split('.', 1)
    return "%s and %s" % (class_string, function_string)


def is_function(string):
    # `True` if it looks like a test function is included in the string.
    return string.startswith('test_') or '.test_' in string


def is_class(string):
    # `True` if first character is uppercase - assume it's a class name.
    return string[0] == string[0].upper()


if __name__ == "__main__":
    if len(sys.argv) > 1:
        pytest_args = sys.argv[1:]
        first_arg = pytest_args[0]

        try:
            pytest_args.remove('--coverage')
        except ValueError:
            pass
        else:
            pytest_args = [
                '--cov', '.',
                '--cov-report', 'xml',
            ] + pytest_args

        if first_arg.startswith('-'):
            # `runtests.py [flags]`
            pytest_args = ['tests'] + pytest_args
        elif is_class(first_arg) and is_function(first_arg):
            # `runtests.py TestCase.test_function [flags]`
            expression = split_class_and_function(first_arg)
            pytest_args = ['tests', '-k', expression] + pytest_args[1:]
        elif is_class(first_arg) or is_function(first_arg):
            # `runtests.py TestCase [flags]`
            # `runtests.py test_function [flags]`
            pytest_args = ['tests', '-k', pytest_args[0]] + pytest_args[1:]
    else:
        pytest_args = []

    sys.exit(pytest.main(pytest_args))
```

```
#!/usr/bin/env python3
import os
import re
import shutil
import sys
from io import open

from setuptools import find_packages, setup

CURRENT_PYTHON = sys.version_info[:2]
REQUIRED_PYTHON = (3, 6)

# This check and everything above must remain compatible with Python 2.7.
if CURRENT_PYTHON < REQUIRED_PYTHON:
    sys.stderr.write("""
==========================
Unsupported Python version
==========================

This version of Django REST Framework requires Python {}.{}, but you're
trying
to install it on Python {}.{}.

This may be because you are using a version of pip that doesn't
understand the python_requires classifier. Make sure you
have pip >= 9.0 and setuptools >= 24.2, then try again:

    $ python -m pip install --upgrade pip setuptools
    $ python -m pip install djangorestframework

This will install the latest version of Django REST Framework which works
on
your version of Python. If you can't upgrade your pip (or Python), request
an older version of Django REST Framework:
```

```python
    $ python -m pip install "djangorestframework<3.10"
""".format(*(REQUIRED_PYTHON + CURRENT_PYTHON)))
    sys.exit(1)


def read(f):
    return open(f, 'r', encoding='utf-8').read()


def get_version(package):
    """
    Return package version as listed in `__version__` in `init.py`.
    """
    init_py = open(os.path.join(package, '__init__.py')).read()
    return re.search("__version__ = ['\"]([^'\"]+)['\"]", init_py).group(1)


version = get_version('rest_framework')


if sys.argv[-1] == 'publish':
    if os.system("pip freeze | grep twine"):
        print("twine not installed.\nUse `pip install twine`.\nExiting.")
        sys.exit()
    os.system("python setup.py sdist bdist_wheel")
    if os.system("twine check dist/*"):
        print("twine check failed. Packages might be outdated.")
        print("Try using `pip install -U twine wheel`.\nExiting.")
        sys.exit()
    os.system("twine upload dist/*")
    print("You probably want to also tag the version now:")
    print("  git tag -a %s -m 'version %s'" % (version, version))
    print("  git push --tags")
    shutil.rmtree('dist')
    shutil.rmtree('build')
    shutil.rmtree('djangorestframework.egg-info')
    sys.exit()


setup(
    name='djangorestframework',
    version=version,
    url='https://www.django-rest-framework.org/',
    license='BSD',
    description='Web APIs for Django, made easy.',
    long_description=read('README.md'),
    long_description_content_type='text/markdown',
    author='Tom Christie',
    author_email='tom@tomchristie.com',  # SEE NOTE BELOW (*)
    packages=find_packages(exclude=['tests*']),
    include_package_data=True,
    install_requires=["django>=2.2", "pytz"],
    python_requires=">=3.6",
    zip_safe=False,
    classifiers=[
        'Development Status :: 5 - Production/Stable',
        'Environment :: Web Environment',
        'Framework :: Django',
        'Framework :: Django :: 2.2',
        'Framework :: Django :: 3.0',
        'Framework :: Django :: 3.1',
        'Framework :: Django :: 3.2',
        'Framework :: Django :: 4.0',
        'Intended Audience :: Developers',
        'License :: OSI Approved :: BSD License',
        'Operating System :: OS Independent',
        'Programming Language :: Python',
        'Programming Language :: Python :: 3',
        'Programming Language :: Python :: 3.6',
        'Programming Language :: Python :: 3.7',
        'Programming Language :: Python :: 3.8',
        'Programming Language :: Python :: 3.9',
        'Programming Language :: Python :: 3.10',
        'Programming Language :: Python :: 3 :: Only',
        'Topic :: Internet :: WWW/HTTP',
    ],
    project_urls={
        'Funding': 'https://fund.django-rest-
framework.org/topics/funding/',
        'Source': 'https://github.com/encode/django-rest-framework',
        'Changelog': 'https://www.django-rest-
framework.org/community/release-notes/',
    },
)
```