Django "quotemanager"

Erik Mayrhofer

Quarkus zu Spring = Flask zu Django

NOTE Homepage: https://www.django-rest-framework.org/

WARNING Trailing slash

Erklärungen

1. Project vs App (Like a Application Server)
2. View, Serializer, Model
3. HATEOAS
4. WSGI

Slide Todos

1. Basic Project
   a. Projekt und App erstellen
   b. Model erstellen (Quote)
   c. Admin panel
   d. Serializer erstellen
   e. View erstellen
   f. URLs eintragen und App installen
   g. Testen
2. Modell verbessern
   a. ModellKlasse
   b. Serializer
   c. View
   d. Url
3. Hateoas
4. Keycloak

Basic Project

Setup

```
pip install django
pip install djangorestframework
```

Create Project

```
django-admin startproject quotemanager

cd quotemanager

python manage.py migrate

python manage.py createsuperuser

charm

python manage.py startapp quotes
```

Modell erstellen

quotes/models.py

```python
from django.db import models


class Quote(models.Model):
    text = models.CharField(max_length=250)
    person = models.CharField(max_length=50)
```

Admin Panel herzeigen

`http://localhost:8000/admin/` im Browser

Serializer erstellen

quotes/serializers.py

```
from rest_framework import serializers

from .models import Quote


class QuoteSerializer(serializers.ModelSerializer):
    class Meta:
        model = Quote
        fields = ('id', 'text', 'person')
```

View erstellen

quotes/views.py

```
from django.shortcuts import render
from rest_framework import viewsets

from .models import Quote
from .serializers import QuoteSerializer


class QuoteView(viewsets.ModelViewSet):
    queryset = Quote.objects.all()
    serializer_class = QuoteSerializer
```

URLs und App eintragen

quotemanager/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('quotes.urls'))
]
```

quotes/urls.py

```python
from django.urls import path, include
from rest_framework import routers

from .views import QuoteView

router = routers.DefaultRouter()
router.register('quotes', QuoteView)

urlpatterns = [
    path('', include(router.urls))
]
```

migrations

```
python manage.py makemigrations

python manage.py migrate

python manage.py runserver
```

Testen

1. `localhost:8000/quotes` im Browser
2. Commandline:

```
http -v POST localhost:8000/quotes/ text="Polymer Teddibär" person="Max Wahl"
```

3. Bei der Console auch Fehlenden Parameter prüfen (validation), und PUT DELETE, zu langer Name (max 50 Zeichen)
4. Admin Gui im Browser `localhost:8000/admin` ⇒ Nichts wird angezeigt

Admin

quotes/admin.py

```python
from django.contrib import admin

from quotes.models import Quote

admin.site.register(Quote)
```

1. Admin Gui im Browser `localhost:8000/admin` ⇒ Anzeige Hässlich

quotes/models.py

```
class Quote(models.Model):
    ...
    def __str__(self):
        return f"{self.text} - {self.person}"
```

Modell Verbessern

Klassen anpassen

Wir fügen eine Person hinzu

quotes/models.py

```
class Person(models.Model): ①
    name = models.CharField(max_length=50)

    def __str__(self):
        return self.name


class Quote(models.Model):
    text = models.CharField(max_length=250)
    person = models.ForeignKey(Person, on_delete=models.CASCADE) ②

    def __str__(self):
        return f"{self.text}" ③
```

① Person erstellen

② Foreign key Setzen

③ Str anpassen

Migrieren

```
python manage.py makemigrations
```

SQLite-DB hat Daten drinnen: Einfach das File löschen

```
python manage.py migrate
```

Serializer

quotes/serializers.py

```
class PersonSerializer(serializers.ModelSerializer):
    class Meta:
        model = Person
        fields = ('id', 'name')
```

View

quotes/views.py

```
class PersonView(viewsets.ModelViewSet):
    queryset = Person.objects.all()
    serializer_class = PersonSerializer
```

Urls

qutoes/urls.py

```
router.register('persons', PersonView)
```

Testen

```
python manage.py runserver
```

1. Web Interface Browsen

Hateoas

quotes/serializers.py

```
class ...Serializer(serializers.HyperlinkedModelSerializer):
    fields = (..., 'url', ....)
```

Keycloak

Install

```
pip install django-oauth-toolkit
```

quotemanager/settings.py

```
INSTALLED_APPS = [
    ...
    'oauth2_provider',
    ...
]

...

OAUTH2_PROVIDER = {
    'SCOPES': {'read': 'Read scope', 'write': 'Write scope', 'groups': 'Access to your
groups'},
    'RESOURCE_SERVER_INTROSPECTION_URL':
        'http://localhost:8080/auth/realms/master/protocol/openid-
connect/token/introspect',
    'RESOURCE_SERVER_INTROSPECTION_CREDENTIALS': ('django-backend', '7031ca56-87dc-
4f2b-aa93-52fb79eb5a86')
}

REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.IsAuthenticated',
    ),
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'oauth2_provider.contrib.rest_framework.OAuth2Authentication',
    ]
}
```

Protecting the API

```
from rest_framework import viewsets, permissions
```

```
class QuoteView(viewsets.ModelViewSet):
    ...
    permission_classes = [permissions.IsAuthenticated]
```

Testing out the Endpoitn

Obtain Bob’s Token

```
http -p b --form \
  -a frontend:460d1a14-b774-482e-b03e-a3830874d9c1 \
  POST localhost:8080/auth/realms/master/protocol/openid-connect/token \
  username=bob password=bob grant_type=password | jq -r ".access_token"

TOKEN=$(http -p b --form \
  -a frontend:460d1a14-b774-482e-b03e-a3830874d9c1 \
  POST localhost:8080/auth/realms/master/protocol/openid-connect/token \
  username=bob password=bob grant_type=password | jq -r ".access_token")
```

Query Protected URL

```
http -v localhost:8000/quotes/ Authorization:"Bearer $TOKEN"
```