Ask Company

리액트와 함께 장고 시작하기 / 장고 DRF

ViewSet과 Router

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

ViewSet

단일 리소스에서 관련있는 View들을 단일 클래스에서 제공

2개의 URL이 필요합니다.

list/create/detail/update/partial_update/delete 등의 멤버 함수로 구현

```python
from rest_framework import viewsets
from rest_framework.response import Response

class PostViewSet(viewsets.ViewSet):
    def list(self, request):
        queryset = Post.objects.all()
        serializer = PostSerializer(queryset, many=True)
        return Response(serializer.data)

    def retrieve(self, request, pk):
        queryset = Post.objects.all()
        user = get_object_or_404(queryset, pk=pk)
        serializer = PostSerializer(user)
        return Response(serializer.data)
```

```python
router = DefaultRouter()
router.register('post', PostViewSet, basename='post')
router.urls
```

https://www.django-rest-framework.org/tutorial/6-viewsets-and-routers/
https://www.django-rest-framework.org/api-guide/routers/

Post 리소스에 대한 2개의 URL

아래 코드 역시 정형화된 패턴

→ ModelViewSet을 통해 간결하게 구현하실 수 있습니다.

```
from rest_framework import generics

class PostListAPIView(generics.ListCreateAPIView):
    queryset = Post.objects.all()
    serializer_class = PostSerializer

class PostDetailAPIView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Post.objects.all()
    serializer_class = PostSerializer
```

ModelViewSet

2가지 ModelViewSet

viewsets.ReadOnlyModelViewSet

- list 지원 → 1개의 URL
- detail 지원 → 1개의 URL

viewsets.ModelViewSet

- list/create 지원 → 1개의 URL
- detail/update/partial_update/delete 지원 → 1개의 URL

URL Patterns에 매핑하기

```python
from rest_framework import viewsets

class PostViewSet(viewsets.ReadOnlyModelViewSet):
    queryset = Post.objects.all()
    serializer_class = PostSerializer
```

개별 View를 만들 수도 있고 …

```python
post_list = PostViewSet.as_view({
    'get': 'list',
})

post_detail = PostViewSet.as_view({
    'get': 'retrieve',
})
```

Router 통해 일괄적으로 urlpatterns에 등록하실 수 있습니다.

```python
from rest_framework.routers import DefaultRouter

router = DefaultRouter()
router.register('post', views.PostViewSet)
                Prefix

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

Router

ReadOnlyModelViewSet과
ModelViewSet에 대해서
동일한 URLPattern 리스트가 생성

- list route
- detail route

```python
from rest_framework.routers import DefaultRouter

router = DefaultRouter()
router.register('post', views.PostViewSet)

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

```
[
    <URLPattern '^post/$' [name='post-list']>,
    <URLPattern '^post\.(?P<format>[a-z0-9]+)/?$' [name='post-list']>,
    <URLPattern '^post/(?P<pk>[^/.]+)/$' [name='post-detail']>,
    <URLPattern '^post/(?P<pk>[^/.]+)\.(?P<format>[a-z0-9]+)/?$' [name='post-detail']>,
    <URLPattern '^$' [name='api-root']>,
    <URLPattern '^\.(?P<format>[a-z0-9]+)/?$' [name='api-root']>
]
```

주의) 구분자가 _ 가 아니라 -

현 Router에 등록된 ViewSet 내역을 조회

ViewSet에 새로운 EndPoint 추가하기 (1/2)

```python
from rest_framework.decorators import action

class PostModelViewSet(viewsets.ModelViewSet):
    queryset = Post.objects.all()
    serializer_class = PostSerializer

    @action(detail=False, methods=['GET'])
    def public(self, request):
        qs = self.queryset.filter(is_public=True)
        serializer = self.get_serializer(qs, many=True)
        return Response(serializer.data)

    @action(detail=True, methods=['PATCH'])
    def set_public(self, request, pk):
        instance = self.get_object()
        instance.is_public = True
        instance.save()
        serializer = self.get_serializer(instance)
        return Response(serializer.data)
```

URL Reverse 명 : basename-함수명 (언더바는 하이픈으로 변경)
즉 post-public 가 됩니다.

ViewSet에 새로운 EndPoint 추가하기 (2/2)

생성된 URL Patterns 리스트

```
[
    ...
    <URLPattern '^post/public/$' [name='post-public']>,
    <URLPattern '^post/public\.(?P<format>[a-z0-9]+)/?$' [name='post-public']>,
    ...
    <URLPattern '^post/(?P<pk>[^/.]+)/set_public/$' [name='post-set-public']>,
    <URLPattern '^post/(?P<pk>[^/.]+)/set_public\.(?P<format>[a-z0-9]+)/?$' [name='post-set-public']>,
    ...
]
```

HTTPie를 활용한 요청의 예

- 쉘> http PATCH http://도메인/post/10/set_public/
- 쉘> http PATCH http://도메인/post/10/ is_public=true

PUT 요청 : 반영할 모든 필드값을 지정
PATCH 요청 : 변경할 값만을 지정

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

-

Ask Company