ViewSet

Django REST framework allows you to combine the logic for a set of related views in a single class, called a ViewSet.

There are two main advantages of using a ViewSet over using a View class.

- Repeated logic can be combined into a single class.
- By using routers, we no longer need to deal with wiring up the URL conf ourselves.

ViewSet Class

A ViewSet class is simply a type of class-based View, that does not provide any method handlers such as get() or post(), and instead provides actions such as list() and create().

- list() – Get All Records.
- retrieve() – Get Single Record
- create() – Create/Insert Record
- update() – Update Record Completely
- partial_update() – Update Record Partially
- destroy() – Delete Record

ViewSet Class

```
from rest_framework import viewsets
class StudentViewSet(viewsets.ViewSet):
    def list(self, request): ……
    def create(self, request):  …….
    def retrieve(self, request, pk=None): ……
    def update(self, request, pk=None): …….
    def partial_update(self, request, pk=None): ……
    def destroy(self, request, pk=None): ……..
```

ViewSet Class

During dispatch, the following attributes are available on the ViewSet:-

- basename - the base to use for the URL names that are created.
- action - the name of the current action (e.g., list, create).
- detail - boolean indicating if the current action is configured for a list or detail view.
- suffix - the display suffix for the viewset type - mirrors the detail attribute.
- name - the display name for the viewset. This argument is mutually exclusive to suffix.
- description - the display description for the individual view of a viewset.

```python
class studentViewset(ViewSet):
    def list(self, request):
        """
        This method wil display all the data in database
        """
        data = student.objects.all()
        serializer = studentSerializer(data, many=True)
        return Response(data=serializer.data)

    def create(self, request):
        """
        This method wil add  data in database
        """
        serializer = studentSerializer(request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(
                data={'msg': 'Your data has been added to database'})
        else:
            return Response(data=serializer.errors)
```

```python
class studentViewset(ViewSet):
    def list(self, request):...

    def create(self, request):...

    def retrieve(self, request, pk=None):
        """
        This method wil get single data from database
        """
        data = student.objects.get(id=pk)
        serializer = studentSerializer(data)
        return Response(data=serializer.data)

    def update(self, request, pk=None):
        """
        This method wil update data in database
        """
        data_to_update = student.objects.get(id=pk)
        serializer = studentSerializer(data_to_update, request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(data={'msg': 'Your data has been updated'})
        else:
            return Response(data=serializer.errors)
```

```python
class studentViewset(ViewSet):
    def list(self, request):...

    def create(self, request):...

    def retrieve(self, request, pk=None):...

    def update(self, request, pk=None):...

    def partial_update(self, request, pk=None):
        """
        This method wil partially update data in database
        """
        data_to_update = student.objects.get(id=pk)
        serializer = studentSerializer(data_to_update,
                                       request.data,
                                       partial=True)
        if serializer.is_valid():
            serializer.save()
            return Response(
                data={'msg': 'Your data has been partially updated'})
        else:
            return Response(data=serializer.errors)
```

```python
class studentViewset(ViewSet):
    def list(self, request):...

    def create(self, request):...

    def retrieve(self, request, pk=None):...

    def update(self, request, pk=None):...

    def partial_update(self, request, pk=None):...

    def destroy(self, request, pk=None):
        """
        This method wil delete data in database
        """
        data = student.objects.get(id=pk)
        data.delete()
        response = {
            'msg': 'data deleted',
        }
        return Response(data=response)
```

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

from . import views
from rest_framework.routers import DefaultRouter

router = DefaultRouter()
router.register('crud',views.studentViewset,'crudViewset')

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

]
```