Django REST Framework-এ **ViewSet** হল এমন একটি শক্তিশালী ফিচার যেটা একাধিক HTTP method (GET, POST, PUT, DELETE)কে এক জায়গায় handle করতে দেয়।
এগুলো ব্যবহার করলে কোড clean, reusable হয় এবং router দিয়ে auto-URL তৈরি করা যায়।

নীচে **সব ধরনের ViewSet**, তাদের ব্যবহার, এবং কখন কোনটা ব্যবহার করবে — A to Z বুঝিয়ে দিলাম।

---

## ✅ 1. `ViewSet` (Base ViewSet)

এটা সবচেয়ে basic ViewSet—এতে সব method manually define করতে হয়।

### ✔ কোথায় ব্যবহার করবো?

- যখন API খুব custom হবে
- প্রতিটি action আলাদা ভাবে define করতে চাই

### ✔ Example:

```
from rest_framework import viewsets

class MyViewSet(viewsets.ViewSet):
    def list(self, request):
        return Response({"msg": "list data"})

    def retrieve(self, request, pk=None):
        return Response({"msg": "single data"})
```

---

## ✅ 2. `ModelViewSet` (Most powerful & most used)

এটা সবচেয়ে বেশি ব্যবহৃত ViewSet।
**Create, Retrieve, Update, Delete, List**—সব নিজে থেকেই তৈরি করে দেয়।

### ✔ কোথায় ব্যবহার করবো?

- যেকোনো Model-এর Full CRUD API বানাতে
- দ্রুত REST API তৈরি করতে

### ✔ Example:

```
from rest_framework import viewsets
from .models import Post
```

```python
from .serializers import PostSerializer

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

**এতেই পুরো CRUD ready 🟩**

---

## ✅ 3. `ReadOnlyModelViewSet` (Only GET allowed)

এতে শুধু `list()` ও `retrieve()` method থাকে।
POST, PUT, DELETE নেই।

### ✔ কখন ব্যবহার করবো?

- ডেটা শুধু দেখাতে চাই
- User access সীমিত
- Public API

### ✔ Example:

```python
from rest_framework import viewsets

class CategoryViewSet(viewsets.ReadOnlyModelViewSet):
    queryset = Category.objects.all()
    serializer_class = CategorySerializer
```

---

## ✅ 4. `GenericViewSet` (Mixins required)

এটা একটি flexible ViewSet।
Mixins দিয়ে যেকোনো combination তৈরি করা যায়।

### ✔ কখন ব্যবহার করবো?

- আপনার প্রয়োজন Full CRUD নয়
- কিছু specific action দরকার
- Custom behavior বানাতে চাই

### ✔ Example (List + Create API only)

```python
from rest_framework import mixins, viewsets
```

```python
class BlogViewSet(mixins.ListModelMixin,
                  mixins.CreateModelMixin,
                  viewsets.GenericViewSet):
    queryset = Blog.objects.all()
    serializer_class = BlogSerializer
```

---

## ✅ 5. `ReadOnlyModelViewSet` vs `ModelViewSet` Differences

| Feature | ModelViewSet | ReadOnlyModelViewSet |
|---|---|---|
| list | ✔ | ✔ |
| retrieve | ✔ | ✔ |
| create | ✔ | ✖ |
| update | ✔ | ✖ |
| delete | ✔ | ✖ |
| Use-case | Full CRUD | Only Read |

---

## 🎯 Kon Somoy Kon ViewSet Use Korbo? (Short & Clear)

| প্রয়োজন | কোন ViewSet বেছে নেব |
|---|---|
| Full CRUD | ✔ `ModelViewSet` |
| শুধুমাত্র ডেটা দেখানো | ✔ `ReadOnlyModelViewSet` |
| কিছু CRUD দরকার (List+Create) | ✔ `GenericViewSet` + Mixins |
| Highly Custom API | ✔ `ViewSet` |
| Complex logic per method | ✔ `ViewSet` / `GenericViewSet` |
| দ্রুত Production ready API | ✔ `ModelViewSet` |

---

## 🎯 Quick Decision Guide:

### যদি model থাকে এবং CRUD দরকার 👉 ModelViewSet

**যদি model থাকে কিন্তু read-only 👉 ReadOnlyModelViewSet**

**যদি CRUD-এর কিছু অংশ দরকার 👉 GenericViewSet + Mixins**

**যদি full custom API 👉 ViewSet**

---

## 🔥 Want Full Code Example?

যদি চান, আমি **4টি ViewSet-এর full project structure**

- **router setup**
- **URL mapping**
- **permissions**
- **custom actions (@action)**

সব সাজিয়ে দিতে পারবো।

---

## example

নিচে **প্রতিটি ViewSet-এর Complete Example** দিলাম—Model, Serializer, ViewSet, URL সব সহ। একবার দেখলেই বুঝে যাবে কোনটা কখন ব্যবহার করতে হয়।

---

## ✅ 📌 Model (একটাই model ব্যবহার করবো সব example এর জন্য)

```python
# models.py
from django.db import models

class Product(models.Model):
    name = models.CharField(max_length=100)
    price = models.DecimalField(max_digits=8, decimal_places=2)
    stock = models.IntegerField(default=0)

    def __str__(self):
        return self.name
```

## ✅ 📌 Serializer

```python
# serializers.py
from rest_framework import serializers
from .models import Product

class ProductSerializer(serializers.ModelSerializer):
    class Meta:
        model = Product
        fields = '__all__'
```

---

## ⭐ 1. Example of `ModelViewSet`

➡ Full CRUD (GET, POST, PUT, PATCH, DELETE)

```python
# views.py
from rest_framework import viewsets
from .models import Product
from .serializers import ProductSerializer

class ProductModelViewSet(viewsets.ModelViewSet):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
```

### URL:

```python
# urls.py
from rest_framework.routers import DefaultRouter
from .views import ProductModelViewSet

router = DefaultRouter()
router.register('products', ProductModelViewSet)

urlpatterns = router.urls
```

---

## ⭐ 2. Example of `ReadOnlyModelViewSet`

➡ শুধু GET (list + retrieve)

```python
# views.py
from rest_framework import viewsets

class ProductReadOnlyViewSet(viewsets.ReadOnlyModelViewSet):
```

```
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
```

### URL:

```
router.register('products-read', ProductReadOnlyViewSet)
```

এখানে POST/PUT/DELETE কাজ করবে না।

---

## ⭐ 3. Example of `GenericViewSet` + Mixins (Custom combination)

➡ শুধু List এবং Create API

```
# views.py
from rest_framework import mixins, viewsets

class ProductListCreateViewSet(mixins.ListModelMixin,
                               mixins.CreateModelMixin,
                               viewsets.GenericViewSet):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
```

### URL:

```
router.register('products-lc', ProductListCreateViewSet)
```

---

## ⭐ 4. Example of Full Custom `ViewSet`

➡ সব method manually handle (Highly custom API)

```
from rest_framework.response import Response
from rest_framework import viewsets, status
from .models import Product
from .serializers import ProductSerializer

class ProductCustomViewSet(viewsets.ViewSet):

    def list(self, request):
        qs = Product.objects.all()
        serializer = ProductSerializer(qs, many=True)
```

```
        return Response(serializer.data)

    def retrieve(self, request, pk=None):
        try:
            product = Product.objects.get(pk=pk)
        except Product.DoesNotExist:
            return Response({"error": "Not Found"}, status=404)

        serializer = ProductSerializer(product)
        return Response(serializer.data)

    def create(self, request):
        serializer = ProductSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=201)
        return Response(serializer.errors, status=400)
```

### URL:

```
router.register('products-custom', ProductCustomViewSet,
basename='products-custom')
```

---

## 🎯 Extra Example: Custom Action in ModelViewSet

➡ Read/Write ViewSet + custom API `/products/low-stock/`

```
from rest_framework.decorators import action
from rest_framework.response import Response

class ProductModelViewSet(viewsets.ModelViewSet):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer

    @action(detail=False, methods=['get'])
    def low_stock(self, request):
        qs = Product.objects.filter(stock__lt=5)
        serializer = ProductSerializer(qs, many=True)
        return Response(serializer.data)
```

### URL:

```
GET /products/low_stock/
```

## 🎯 কোন ViewSet কখন ব্যবহার করবো? (Example Based)

| Scenario | Best ViewSet | Example |
|---|---|---|
| Full CRUD | `ModelViewSet` | ProductModelViewSet |
| Public, Read-only API | `ReadOnlyModelViewSet` | ProductReadOnlyViewSet |
| শুধু List + Create | `GenericViewSet + mixins` | ProductListCreateViewSet |
| Fully Custom Logic | `ViewSet` | ProductCustomViewSet |
| Default CRUD + Extra endpoints | `ModelViewSet + action()` | low_stock action |

## 🔥 চাইলে আমি এগুলোর উপর ভিত্তি করে

**User, Course, Blog, Job, E-commerce**
—সব কিছুর FULL API structure create করে দিতে পারবো।

## নি, এবার আমি একই Model ব্যবহার করে

**৪ রকম ভাবে API তৈরি করা** দেখাচ্ছি:

1️⃣ **APIView দিয়ে**
2️⃣ **ViewSet দিয়ে**
3️⃣ **GenericViewSet + Mixins দিয়ে**
4️⃣ **ModelViewSet দিয়ে**

👉 একই কাজ—কিন্তু ৪ রকম style
এভাবে দেখলে আপনি যেকোনো style use করতে পারবেন।

## 📌 Model (common)

```python
# models.py
from django.db import models

class Student(models.Model):
    name = models.CharField(max_length=100)
    age = models.IntegerField()
```

```
    course = models.CharField(max_length=100)

    def __str__(self):
        return self.name
```

## 📌 Serializer

```
# serializers.py
from rest_framework import serializers
from .models import Student

class StudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = '__all__'
```

---

## ✅ 1. APIView — Full Manual Control (Basic to Advanced)

DRF-এ সবচেয়ে custom API।

```
# views.py
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import Student
from .serializers import StudentSerializer

class StudentAPIView(APIView):

    def get(self, request):
        students = Student.objects.all()
        serializer = StudentSerializer(students, many=True)
        return Response(serializer.data)

    def post(self, request):
        serializer = StudentSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,
status=status.HTTP_201_CREATED)
        return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)
```

### 👉 URL:

```
path('students/', StudentAPIView.as_view()),
```

**➡ এখানে সব manually লিখতে হয়।**

---

## ✅ 2. ViewSet — Manual + Router Auto URL

APIView-এর মত, কিন্তু router URL automatically create করে।

```
# views.py
from rest_framework import viewsets

class StudentViewSet(viewsets.ViewSet):

    def list(self, request):
        students = Student.objects.all()
        serializer = StudentSerializer(students, many=True)
        return Response(serializer.data)

    def retrieve(self, request, pk=None):
        student = Student.objects.get(pk=pk)
        serializer = StudentSerializer(student)
        return Response(serializer.data)

    def create(self, request):
        serializer = StudentSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors)
```

### 👉 URL (router)

```
from rest_framework.routers import DefaultRouter
router = DefaultRouter()
router.register("students", StudentViewSet, basename="students")
urlpatterns = router.urls
```

➡ `list`, `retrieve`, `create` manually লিখতে হয়।
➡ APIView-এর মতোই, কিন্তু URLs auto generate হয়।

---

## ✅ 3. GenericViewSet + Mixins — Half Automatic

List + Create + Retrieve চাইলে mixins ব্যবহার করো।

```
# views.py
from rest_framework import viewsets, mixins

class StudentGenericViewSet(mixins.ListModelMixin,
                            mixins.CreateModelMixin,
                            mixins.RetrieveModelMixin,
                            viewsets.GenericViewSet):

    queryset = Student.objects.all()
    serializer_class = StudentSerializer
```

### 👉 URL (router)

```
router.register("students-generic", StudentGenericViewSet)
```

➡ list, retrieve, create auto handled by mixins
➡ খুব কম কোডে কাজ হয়ে যায়।

---

## ✅ 4. ModelViewSet — Everything Automatic (Most Powerful)

Full CRUD + router auto URL + less code.

```
# views.py
from rest_framework import viewsets

class StudentModelViewSet(viewsets.ModelViewSet):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
```

### 👉 URL:

```
router.register("students-model", StudentModelViewSet)
```

➡ list(), retrieve(), create(), update(), delete() সব auto ready.
➡ সবচেয়ে বেশি ব্যবহার হয়।

---

# 🎯 Final Comparison Table

| Method | Code Amount | Control | URL Handling | Use Case |
|---|---|---|---|---|
| **APIView** | বেশি | Full manual | Manual path | Custom API |
| **ViewSet** | মধ্যম | Manual methods | Router auto | Custom but grouped |
| **GenericViewSet + Mixins** | কম | মাঝারি | Router auto | Semi-custom CRUD |
| **ModelViewSet** | সবচেয়ে কম | High-level | Router auto | Full CRUD |

---

# 🧪 Final: এক API — ৪ভাবে Calls (for practice)

## 1️⃣ APIView

```
GET /students/
POST /students/
```

## 2️⃣ ViewSet

```
GET /students/
GET /students/1/
POST /students/
```

## 3️⃣ GenericViewSet + Mixins

same as ViewSet:

```
GET /students-generic/
POST /students-generic/
GET /students-generic/1/
```

## 4️⃣ ModelViewSet

Full CRUD

```
GET /students-model/
GET /students-model/1/
POST /students-model/
```

```
PUT /students-model/1/
DELETE /students-model/1/
```

---

## 🔥 চাইলে আমি

👉 একই কাজ Class-based Generic Views দিয়ে
👉 Function-Based Views দিয়ে
👉 Custom action সহ ViewSet ( @action )
_____

# ঠিক আছে! এখন আমি একই Student API–কে ৩ ভাবে দেখাবো:

---

## ✅ 1. Class-Based Generic Views

(**ListAPIView**, **CreateAPIView**, **RetrieveAPIView**, **UpdateAPIView**, **DestroyAPIView**)
সব এক এক করে দেখাবো।

### 📌 List + Create

```
from rest_framework import generics
from .models import Student
from .serializers import StudentSerializer

class StudentListCreateView(generics.ListCreateAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
```

### 📌 Retrieve + Update + Delete

```
class StudentDetailView(generics.RetrieveUpdateDestroyAPIView):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer
```

### 👉 URLs

```
path('students/', StudentListCreateView.as_view()),
path('students/<int:pk>/', StudentDetailView.as_view()),
```

➡ DRF Generic Views সবচেয়ে কম কোডে powerful CRUD দেয়।

---

# ✅ 2. Function-Based Views (FBV) with @api_view

Fully manual, but easy to understand.

```
from rest_framework.decorators import api_view
from rest_framework.response import Response
from rest_framework import status
from .models import Student
from .serializers import StudentSerializer

@api_view(['GET', 'POST'])
def student_list_create(request):

    if request.method == 'GET':
        students = Student.objects.all()
        serializer = StudentSerializer(students, many=True)
        return Response(serializer.data)

    elif request.method == 'POST':
        serializer = StudentSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,
status=status.HTTP_201_CREATED)
        return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)


@api_view(['GET', 'PUT', 'DELETE'])
def student_detail(request, pk):

    try:
        student = Student.objects.get(pk=pk)
    except Student.DoesNotExist:
        return Response({"error": "Student not found"}, status=404)

    if request.method == 'GET':
        serializer = StudentSerializer(student)
        return Response(serializer.data)

    elif request.method == 'PUT':
        serializer = StudentSerializer(student, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors)

    elif request.method == 'DELETE':
```

```
        student.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

### 👉 URLs

```
path('students-fbv/', student_list_create),
path('students-fbv/<int:pk>/', student_detail),
```

➡ FBV = বেশি কোড কিন্তু pure logic control।

---

## ✅ 3. Custom Action with ViewSet ( @action )

এখানে আমরা ModelViewSet ব্যবহার করবো।

### 📌 Use Case: Extra API

যেমন:
✔ low stock
✔ search
✔ filter
✔ recent added
✔ custom logic

Example:
**GET /students-model/search/?name=rahim**

```
from rest_framework.decorators import action
from rest_framework import viewsets
from rest_framework.response import Response
from .models import Student
from .serializers import StudentSerializer

class StudentModelViewSet(viewsets.ModelViewSet):
    queryset = Student.objects.all()
    serializer_class = StudentSerializer

    @action(detail=False, methods=['get'])
    def search(self, request):
        name = request.query_params.get('name', None)
        if name:
            qs = Student.objects.filter(name__icontains=name)
        else:
            qs = Student.objects.none()

        serializer = StudentSerializer(qs, many=True)
```

```
        return Response(serializer.data)

    @action(detail=True, methods=['get'])
    def info(self, request, pk=None):
        """Detail-based custom action"""
        student = self.get_object()
        return Response({
            "id": student.id,
            "name": student.name,
            "age": student.age,
            "course": student.course,
            "msg": "This is custom detail action."
        })
```

### 👉 URLs (router দিয়ে):

```
router.register('students-model', StudentModelViewSet)
```

### 🔥 Resulting endpoints:

| Endpoint | Action type |
|---|---|
| `/students-model/` | list |
| `/students-model/1/` | retrieve |
| `/students-model/search/?name=ali` | custom (list-style) |
| `/students-model/1/info/` | custom (detail-style) |

---

## 🎯 Summary — এখন আপনি ৬টি style শিখলেন

| Style | Type | Flexibility | Code amount | Beginner friendly |
|---|---|---|---|---|
| **APIView** | Manual | High | Medium | Medium |
| **ViewSet** | Manual + router | High | Medium | Medium |
| **GenericViewSet + mixins** | Semi-automatic | Medium | Low | Easy |
| **ModelViewSet** | Fully automatic | Medium | Very low | Easiest |
| **Generic APIViews** | Automatic CRUD | Medium | Low | Very easy |

| Style | Type | Flexibility | Code amount | Beginner friendly |
|---|---|---|---|---|
| **FBV** | Fully manual | Very high | Highest | Good for beginners |

---

## 👉 চাইলে আমি এগুলোকে

✔ Authentication
✔ Permission classes
✔ Pagination
✔ Filtering, Search
✔ Ordering
✔ Throttling
—এসব apply করে complete production API বানিয়ে দিতে পারি।

বলুন কোনটা চান?