DRF Class-Based Views - Detailed Notes

```
Django REST Framework (DRF) - Class-Based Views Detailed Notes
================================================================

1. Base View Classes
--------------------
APIView
  - Base class for all DRF views.
  - Provides request parsing, authentication, permissions.

GenericAPIView
  - Extends APIView.
  - Adds queryset, serializer_class, lookup_field, pagination.

Mixins (must be combined with GenericAPIView or GenericViewSet)
  ListModelMixin       -> GET list
  CreateModelMixin     -> POST create
  RetrieveModelMixin   -> GET retrieve
  UpdateModelMixin     -> PUT/PATCH update
  DestroyModelMixin    -> DELETE destroy

2. Concrete Generic Views (Single-Purpose)
-----------------------------------------
ListAPIView                    -> GET list
CreateAPIView                  -> POST create
RetrieveAPIView                -> GET retrieve
UpdateAPIView                  -> PUT/PATCH update
DestroyAPIView                 -> DELETE destroy

3. Combined Generic Views (Common Combos)
-----------------------------------------
ListCreateAPIView              -> GET list, POST create
RetrieveUpdateAPIView          -> GET retrieve, PUT/PATCH update
RetrieveDestroyAPIView         -> GET retrieve, DELETE destroy
RetrieveUpdateDestroyAPIView   -> GET retrieve, PUT/PATCH update, DELETE destroy

Example URL patterns:
from django.urls import path
from .views import StudentListCreate, StudentDetail

urlpatterns = [
    path('students/', StudentListCreate.as_view(), name='student-list'),
    path('students/<int:pk>/', StudentDetail.as_view(), name='student-detail'),
]

4. ViewSets
-----------
ViewSet                 -> Base class for defining multiple actions.
GenericViewSet          -> Adds GenericAPIView features.
ReadOnlyModelViewSet    -> list, retrieve
ModelViewSet            -> list, create, retrieve, update, partial_update, destroy
```

DRF Class-Based Views - Detailed Notes

```
Mixins + GenericViewSet examples:
ListModelMixin + GenericViewSet          -> list only
RetrieveModelMixin + GenericViewSet      -> retrieve only

Routers:
from rest_framework.routers import DefaultRouter
router = DefaultRouter()
router.register('students', StudentViewSet)

Automatic endpoints for ModelViewSet:
GET    /students/          -> list
POST   /students/          -> create
GET    /students/{pk}/     -> retrieve
PUT    /students/{pk}/     -> update
PATCH  /students/{pk}/     -> partial_update
DELETE /students/{pk}/     -> destroy

Custom actions (@action decorator):
from rest_framework.decorators import action

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

    @action(detail=True, methods=['post'])
    def enroll(self, request, pk=None):
        # custom logic
        return Response({'status': 'enrolled'})

Generates endpoint POST /students/{pk}/enroll/

5. Choosing the Right View
--------------------------
Need only read operations       -> ReadOnlyModelViewSet or ListAPIView/RetrieveAPIView
Need full CRUD                  -> ModelViewSet or ListCreate/RetrieveUpdateDestroy
Custom logic per HTTP method    -> APIView
Complex endpoint without router -> GenericAPIView + Mixins

Summary
-------
DRF offers multiple levels of abstraction:
- APIView for full control.
- GenericAPIView + Mixins for modular actions.
- Concrete Generic Views for common patterns.
- ViewSets + Routers for automatic CRUD endpoints.

Use these tools to build maintainable, scalable REST APIs quickly.
```