class GenericAPIView:

--> **drf.generics.py**

class APIView:

- .authentication_classes
- .permission_classes
- get_permissions()
- renderer, parserers props

- queryset = None
- serializer_class = None
- lookup_field='pk'
- pagination_class = default
- get_queryset()
- get_object()
- get_serializer()
- get_serializer_class()
- get_serializer_context()
- filter_queryset()
- paginate_queryset()

Mixins

 # --> drf.mixins.py

- class **CreateModelMixins**: create(), perform_create()
- class **ListModelMixins**: list()
- class **RetrieveModelMixin**: retrieve()
- class **UpdateModelmixin**: update(), perform_update(), partial_update()
- class **DestroyModelMixin**: destroy(), perform_destroy()

APIViews

Concrete view classes that provide method handlers by composing the mixin classes with the base view.
--> **drf.generics.py**

```
class CreateAPIView(mixins.CreateModelMixin, GenericAPIView):
    def post(self, request, *args, **kwargs):
        return self.create(request, *args, **kwargs)


class ListAPIView(mixins.ListModelMixin, GenericAPIView):
    def get(self, request, *args, **kwargs):
        return self.list(request, *args, **kwargs)


class RetrieveAPIView(mixins.RetrieveModelMixin, GenericAPIView):
    def get(self, request, *args, **kwargs):
        return self.retrieve(request, *args, **kwargs)


class DestroyAPIView(mixins.DestroyModelMixin, GenericAPIView):
    def delete(self, request, *args, **kwargs):
        return self.destroy(request, *args, **kwargs)


class UpdateAPIView(mixins.UpdateModelMixin, GenericAPIView):
    def put(self, request, *args, **kwargs):
        return self.update(request, *args, **kwargs)

    def patch(self, request, *args, **kwargs):
        return self.partial_update(request, *args, **kwargs)


class ListCreateAPIView(mixins.ListModelMixin, mixins.CreateModelMixin, GenericAPIView):
    def get(self, request, *args, **kwargs):
        return self.list(request, *args, **kwargs)

    def post(self, request, *args, **kwargs):
        return self.create(request, *args, **kwargs)


class RetrieveUpdateAPIView(mixins.RetrieveModelMixin, mixins.UpdateModelMixin, GenericAPIView):
    def get(self, request, *args, **kwargs):
        return self.retrieve(request, *args, **kwargs)

    def put(self, request, *args, **kwargs):
        return self.update(request, *args, **kwargs)

    def patch(self, request, *args, **kwargs):
```

```
        return self.partial_update(request, *args, **kwargs)


class RetrieveDestroyAPIView(mixins.RetrieveModelMixin,  mixins.DestroyModelMixin,  GenericAPIView):
    def get(self, request, *args, **kwargs):
        return self.retrieve(request, *args, **kwargs)

    def delete(self, request, *args, **kwargs):
        return self.destroy(request, *args, **kwargs)


class RetrieveUpdateDestroyAPIView(mixins.RetrieveModelMixin,mixins.UpdateModelMixin,
                                   mixins.DestroyModelMixin, GenericAPIView):
    def get(self, request, *args, **kwargs):
        return self.retrieve(request, *args, **kwargs)

    def put(self, request, *args, **kwargs):
        return self.update(request, *args, **kwargs)

    def patch(self, request, *args, **kwargs):
        return self.partial_update(request, *args, **kwargs)

    def delete(self, request, *args, **kwargs):
        return self.destroy(request, *args, **kwargs)
```

ViewSets

ViewSets are essentially just a type of class based view, that doesn't provide any method handlers, such as `get()`, `post()`, etc... but instead has actions, such as `list()`, `retrieve()`, `create()`, etc... (--> in **drf.viewsets.py**)

class **ViewSetMixin**:

- This is magic
- Overrides `.as_view()` so that it takes an `actions` keyword that performs the binding of HTTP methods to actions on the Resource. For example, to create a concrete view binding the 'GET' and 'POST' methods to the 'list' and 'create' actions...
- def reverse_action()
  - Reverse the action for the given `url_name`
- def **get_extra_actions**():
  - Get the methods that are marked as an extra ViewSet `@action`.

```
class ViewSet(ViewSetMixin, views.APIView):

    # The base ViewSet class does not provide any actions by default.

    pass


class GenericViewSet(ViewSetMixin, generics.GenericAPIView):

    """  The GenericViewSet class does not provide any actions by default, but does include the base set of generic view
        behavior, such as the `get_object` and `get_queryset` methods."""

    pass

class ReadOnlyModelViewSet(mixins.RetrieveModelMixin, mixins.ListModelMixin, GenericViewSet):

    """ A viewset that provides default `list()` and `retrieve()` actions. """

    pass

class ModelViewSet(mixins.CreateModelMixin, mixins.RetrieveModelMixin,mixins.UpdateModelMixin,

      mixins.DestroyModelMixin,mixins.ListModelMixin,GenericViewSet):

    """A viewset that provides default `create()`, `retrieve()`, `update()`,()`, `destroy()` and `list()` actions."""

    pass
```