"""

#Generic View Set

class PassportViewSet(viewsets.GenericViewSet,

                      mixins.ListModelMixin,

                      mixins.CreateModelMixin,

                      mixins.RetrieveModelMixin,

                      mixins.DestroyModelMixin):

    queryset = PassportDatabase.objects.all()

    serializer\_class = PassportSerializer

"""

"""

#VIEWSETS

class PassportViewSet(viewsets.ViewSet):

    def list(self, request):

        passport = PassportDatabase.objects.all()

        serializer = PassportSerializer(passport,many=True)

        return Response(serializer.data)

    def create(self, request):

        serializer = PassportSerializer(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)

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

        queryset = PassportDatabase.objects.all()

        passport = get\_object\_or\_404(queryset,pk=pk)

        serializer = PassportSerializer(passport)

        return Response(serializer.data)

    def update(self,request,pk=None):

        passport = PassportDatabase.objects.get(pk=pk)

        serializer = PassportSerializer(passport,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)

    def destroy(self,request,pk=None):

        passport = PassportDatabase.objects.get(pk=pk)

        passport.delete()

        return Response(status=status.HTTP\_204\_NO\_CONTENT)

"""

"""

The mixin classes provide the actions that are used to provide the basic view behavior. Note that the mixin classes provide action methods rather than defining the handler methods, such as .get() and .post(), directly. This allows for more flexible composition of behavior.

"""

"""

#Mixin views

class Passport\_List(generics.GenericAPIView,

                    mixins.ListModelMixin,

                    mixins.CreateModelMixin):

    queryset = PassportDatabase.objects.all()

    serializer\_class = PassportSerializer

    #ListModelMixin is used to show/get the items

    def get(self,request):

        return self.list(request)

    #CreateModelMixin is used to create/post the items

    def post(self,request):

        return self.create(request)

class Passport\_Details(generics.GenericAPIView,

                    mixins.RetrieveModelMixin,

                    mixins.UpdateModelMixin,

                    mixins.DestroyModelMixin):

                    queryset = PassportDatabase.objects.all()

                    serializer\_class = PassportSerializer

                    lookup\_field = 'id'

                    def get(self,request,id):

                        return self.retrieve(request,id=id)

                    def put(self,request,id):

                        return self.update(request,id=id)

                    def delete(self,request,id):

                        return self.destroy(request,id=id)

"""

"""

#class based api VIEWS

class Passport\_List(APIView):

    def get(self, request):

        passport = PassportDatabase.objects.all()

        serializer = PassportSerializer(passport,many=True)

        return Response(serializer.data)

    def post(self, request):

        serializer = PassportSerializer(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)

class Passport\_Details(APIView):

    def get\_object(self, id):

        try:

            return PassportDatabase.objects.get(id=id)

        except PassportDatabase.DoesNotExist:

            return Response(status=status.HTTP\_404\_NOT\_FOUND)

    def get(self,request,id):

        passport = self.get\_object(id)

        serializer = PassportSerializer(passport)

        return Response(serializer.data)

    def put(self,request,id):

        passport = self.get\_object(id)

        serializer = PassportSerializer(passport,data=request.data)

        if serializer.is\_valid():

            serializer.save()

            return Response(serializer.data)

        return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

    def delete(self,request,id):

        passport = self.get\_object(id)

        passport.delete()

        return Response(status=status.HTTP\_204\_NO\_CONTENT)

"""

"""FUNCTION BASED API VIEWS

# @csrf\_exempt

@api\_view(['GET','POST'])  #THIS GIVES AN API INTERFACE

def Passport\_List(request):

    if request.method == 'GET':

        passport = PassportDatabase.objects.all()

        serializer = PassportSerializer(passport,many=True)

        return Response(serializer.data)

        # return JsonResponse(serializer.data, safe=False)

    elif request.method == 'POST':

        # data = JSONParser().parse(request)

        serializer = PassportSerializer(data=request.data)

        if serializer.is\_valid():

            serializer.save()

            return Response(serializer.data, status=status.HTTP\_201\_CREATED)

            # return JsonResponse(serializer.data, status=201)

        return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

# @csrf\_exempt

@api\_view(['GET','PUT','DELETE'])

def Passport\_Details(request,pk):

    #pk is primary key

    try:

        passport = PassportDatabase.objects.get(pk=pk)

    except PassportDatabase.DoesNotExist:

        return Response(status=status.HTTP\_404\_NOT\_FOUND)

    if request.method == 'GET':

        serializer = PassportSerializer(passport)

        return Response(serializer.data)

    elif request.method == 'PUT':

        # data = JSONParser().parse(request)

        serializer = PassportSerializer(passport,data=request.data)

        if serializer.is\_valid():

            serializer.save()

            return Response(serializer.data)

        return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

    elif request.method == 'DELETE':

        passport.delete()

        return Response(status=status.HTTP\_204\_NO\_CONTENT)

"""