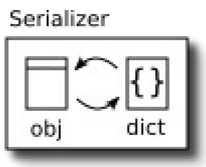
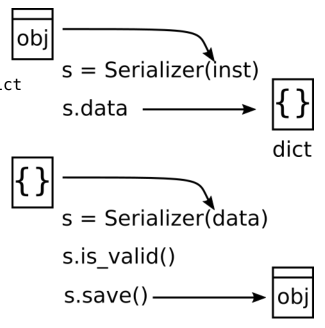
Django DRF Cheat Sheet





Serialize:
```
  # Object instance -> Data dict
  s = Serializer(instance)
  s.data
```

Deserialize (create):
```
  # dict -> Object
  s = Serializer(data={..})
  s.is_valid()
  instance = s.save()
```

Validation:
```
   s = Serializer(data={..})
  if s.is_valid(raise_exception=False):
    s.validated_data
  else:
    s.errors  # dict(field=[problems..])
```

Update:
```
  s = Serializer(instance, validated_data, partial=True)
  new = s.save()
```

Serializer Classes:
```
  class SimpleSerializer(serializers.Serializer):
    name = serializers.CharField(max_length=128)
    user = UserSerializer(required=false)  # nested
    ...

  def create(self, validated_data):
    kwargs = self.context['view'].kwargs
    ...
    return instance

  def update(self, instance, validated_data):
    # if self.partial
    ...
    return instance

  def save(self, **kwargs):
    # instance = super().save(**kwargs)
    nm = self.validated_data.get('name')
    ...
    instance = model_object.save()
    return instance

  # def validate_<field>(self, name):
  def validate_name(self, name):
    if not name:
      raise serializers.ValidationError('Name required')
    ...
    return name

  def validate(self, data):
    nm = data.get('name')
    if not nm:
      raise serializers.ValidationError({'name': ..})
    ...
    return validated_data

  class DbSerializer(serializers.ModelSerializer):
    class Meta:
      model = DbModel
      fields = [..]
      list_serializer_class = DbListSerializer
    def to_representation(self, obj) → Dict
    def to_internal(self, data) → Dict # validated
```

Create	Serializer(data={..}); s.save(): Object
Serialize	Serializer(instance); s.data: Dict
Update	Serializer(instance, data={..}, partial=True)
Set	Serializer(instance, data={..})

Serializer Fields:

Common: CharField, UUIDField, **Null**BooleanField, DateTimefield, JSONField, EmailField, ..

Special: SerializerMethodField, RelatedField, PrimaryKeyRelatedField, SlugRelatedField, HyperlinkedRelatedField..

Field Args

	Default	Input	Output	Notes
read_only	False	no	yes	
write_only	False	yes	no	
required	True	yes	yes	raise if missing
allow_null	False			None is allowed
default	<value>	sets		If set, don't use the arg "required"
source	the python form of field access like "user.address.zip" (not double underscore) OR a callable name on the object like get_primary_phone()			

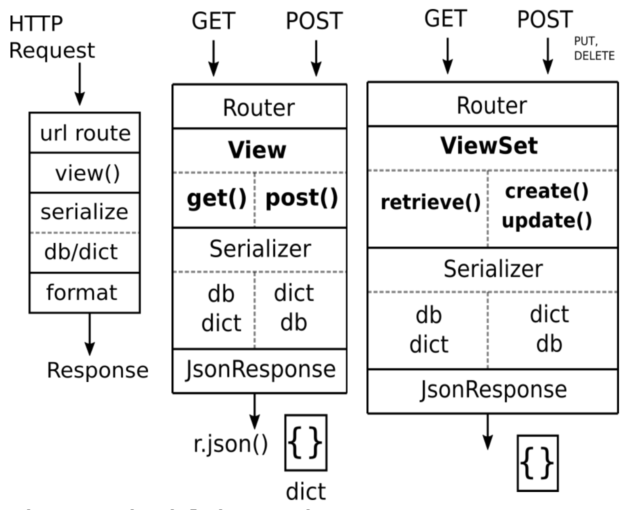
Request
```
- request.data - dict of (post, put, patch)
- request.query_param (get)
- request.user (django.contrib.auth.models.AnonymousUser)
- request.auth (usually a model instance)
- HTTP stuff: request.method (e.g. GET), .content_type (e.g.
   application/json), .META, .headers, .path
```

view() functions
```
# urls.py
urlpatterns = [
  path(r'viewfn/<int:arg1>/', views.my_view_fn),
]

# views.py
from django.http import JsonResponse
...
  def my_view_fn(request, arg1=None):
    if request.method == 'POST':
      return JsonResponse({}, status=201)
```

Testing
```
from rest_framework.test import APIClient, APITestCase

class MyTests(APITestCase):
  def test_basics(self):
    client = APIClient()
    response = client.post(f'/viewtest/{arg1}', data={..})
    assert response.status_code == 200
    assert response.json().get('name') == name
```

Views
```
  class SimpleView(APIView):
    def post(self, request, *args, **kwargs):
      request.data.get('name')
      return Response(..)
    def get(self, request, *args, **kwargs):
      request.query_params.get('name')
      ...
    def patch(self, request, *args, **kwargs):
      ...
    def delete(self, request, *args, **kwargs):
      ...
```

http://thayer.b2si.com/drf.html constructive feedback welcome



ViewSets (ModelViewSets)
```
from rest_framework import permissions

  class SimpleViewSet(ViewSet):  # ModelViewSet
    serializer_class = SimpleSerializer
    # queryset = SimpleModel.objects.all()
    lookup_field = 'field'
    authentication_classes = (..,)
    permissions_classes = (permissions.IsAuthenticated,)
    # permissions_classes = (permissions.AllowAny,)
    # filter_backends = (..,)

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

    def update(self, request, *args, **kwargs):
    def partial_update(self, request, *args, **kwargs):
    def destroy(self, request, *args, **kwargs):
    def retrieve(self, request, *args, **kwargs):
    def list(self, request, *args, **kwargs):

    def get_queryset(self):
        # related to self.queryset
        data = self.request.data
        qp = self.request.query_params
        kwargs = self.kwargs  # args for view

    def get_object(self): → instance
        # see self.lookup_field

    def get_serializer(self):
    def get_serializer_context(self): → dict(request=, view=
      …

    @detail_router(['GET', 'POST'])
    def custom(request, *args, **kwargs):

# urls.py
from rest_framework.routers import DefaultRouter

router = DefaultRouter()
router.register(
    r'viewset', example.SimpleViewSet, basename='Simple')

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

What Goes Here?