Django REST Framework (DRF) Cheat Sheet

1. Installation & Setup

Simple definition: How to install DRF and add it to your Django project.

```
pip install djangorestframework
```

Add to `settings.py`:

```
INSTALLED_APPS = [
    'rest_framework',
]
```

---

2. Basic Serializer

Simple definition: Converts Python objects to JSON and validates incoming data.

```
from rest_framework import serializers

class UserSerializer(serializers.Serializer):
    name = serializers.CharField(max_length=100)
    age = serializers.IntegerField()
```

3. ModelSerializer

Simple definition: A serializer that automatically maps Django models to API fields.

```
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = '__all__'
```

---

4. Views

Simple definition: The part that receives requests and returns responses.

APIView

```python
from rest_framework.views import APIView
from rest_framework.response import Response

class Hello(APIView):
    def get(self, request):
        return Response({"msg": "Hello"})
```

Generic Views

```python
from rest_framework import generics

class UserListCreate(generics.ListCreateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
```

ModelViewSet

```python
from rest_framework import viewsets

class UserViewSet(viewsets.ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer
```

---

5. Routers

Simple definition: Automatically creates URL routes for ViewSets.

```python
from rest_framework.routers import DefaultRouter
from .views import UserViewSet

router = DefaultRouter()
router.register(r'users', UserViewSet)

urlpatterns = router.urls
```

---

6. Query Parameters

Simple definition: Extra data passed in the URL to filter or modify results.

```
request.query_params.get('page')
```

---

7. POST Data

Simple definition: How you accept and validate data from API requests.

```
serializer = UserSerializer(data=request.data)
if serializer.is_valid():
    serializer.save()
```

---

8. Response

Simple definition: The object used to send data back to the client.

```
from rest_framework.response import Response
return Response({"status": "ok"}, status=200)
```

---

9. Status Codes

Simple definition: Standard HTTP codes that show if a request succeeded or failed.

```
from rest_framework import status
return Response(status=status.HTTP_201_CREATED)
```

---

10. Permissions

Simple definition: Controls who can access an API endpoint.

```
from rest_framework.permissions import IsAuthenticated

class View(APIView):
    permission_classes = [IsAuthenticated]
```

---

11. Authentication Types

Simple definition: How users prove their identity to your API.

```
Token Authentication
Session Authentication
JWT Authentication
```

JWT Setup (SimpleJWT)

```
pip install djangorestframework-simplejwt
```

Add endpoints:

```
from rest_framework_simplejwt.views import (
    TokenObtainPairView,
    TokenRefreshView,
)
urlpatterns = [
    path('token/', TokenObtainPairView.as_view()),
    path('refresh/', TokenRefreshView.as_view()),
]
```

---

12. Pagination

Simple definition: Splitting large results into smaller pages.

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS':
'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 10,
}
```

---

13. Filtering & Searching

Simple definition: Allows searching and ordering data in API responses.

Add to settings

```
'DEFAULT_FILTER_BACKENDS': [
    'rest_framework.filters.SearchFilter',
    'rest_framework.filters.OrderingFilter',
]
```

Use in view

```
search_fields = ['name', 'email']
ordering_fields = ['age']
```

---

14. Custom Validation

Simple definition: Adding extra rules to control what data is allowed.

```
class UserSerializer(serializers.ModelSerializer):
    def validate_age(self, age):
        if age < 0:
            raise serializers.ValidationError("Age cannot be negative")
        return age
```

---

15. Overriding create/update

Simple definition: Customizing how objects are saved or updated.

```
class UserSerializer(serializers.ModelSerializer):
    def create(self, data):
        return User.objects.create(**data)

    def update(self, instance, data):
        instance.name = data.get('name', instance.name)
        instance.save()
        return instance
```

---

16. File Uploads

Simple definition: Handling images, PDFs, and other file inputs.

```
class FileSerializer(serializers.Serializer):
    file = serializers.FileField()
```

---

17. Throttling

Simple definition: Limits how many requests a user can make.

```
REST_FRAMEWORK = {
    'DEFAULT_THROTTLE_CLASSES': [
        'rest_framework.throttling.UserRateThrottle',
    ],
    'DEFAULT_THROTTLE_RATES': {
        'user': '100/hour',
    }
}
```

---

18. Common Errors

Simple definition: Frequent issues and what they mean in API development.

```
415 Unsupported Media Type → missing JSON header
400 Bad Request → invalid serializer
403 Forbidden → permission denied
401 Unauthorized → missing token
```

---

19. Best Practices

Simple definition: Recommended methods for building clean, secure APIs.

```
Use ModelSerializer
Always validate data
Use ViewSets + Routers
Use JWT for production auth
Use pagination for large lists
```