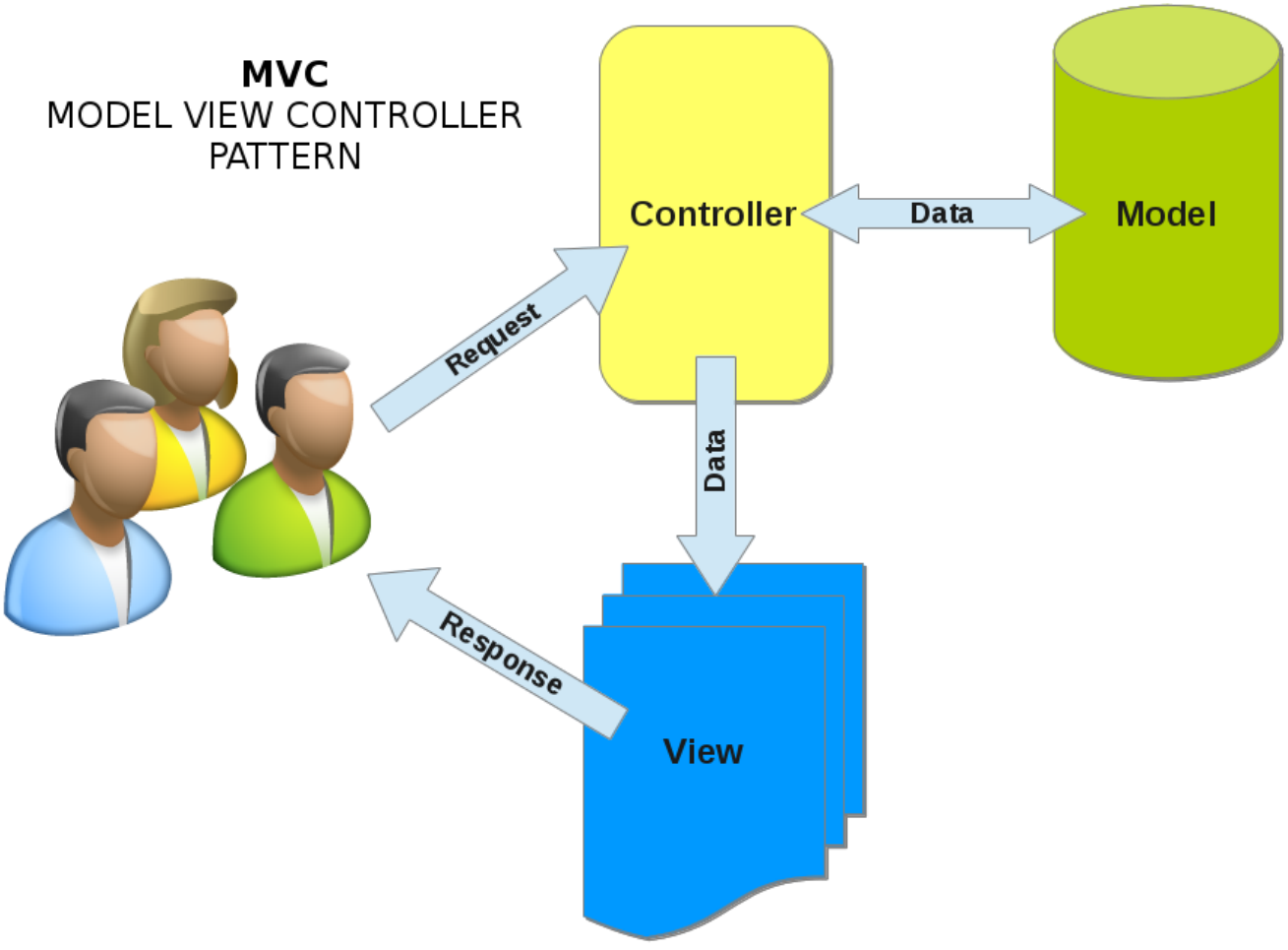
DRF

@pvavilin

28 июня 2023 г.

Outline

MVC


MVC
MODEL VIEW CONTROLLER
PATTERN
Controller
Data
Model
Request
Data
Response
View

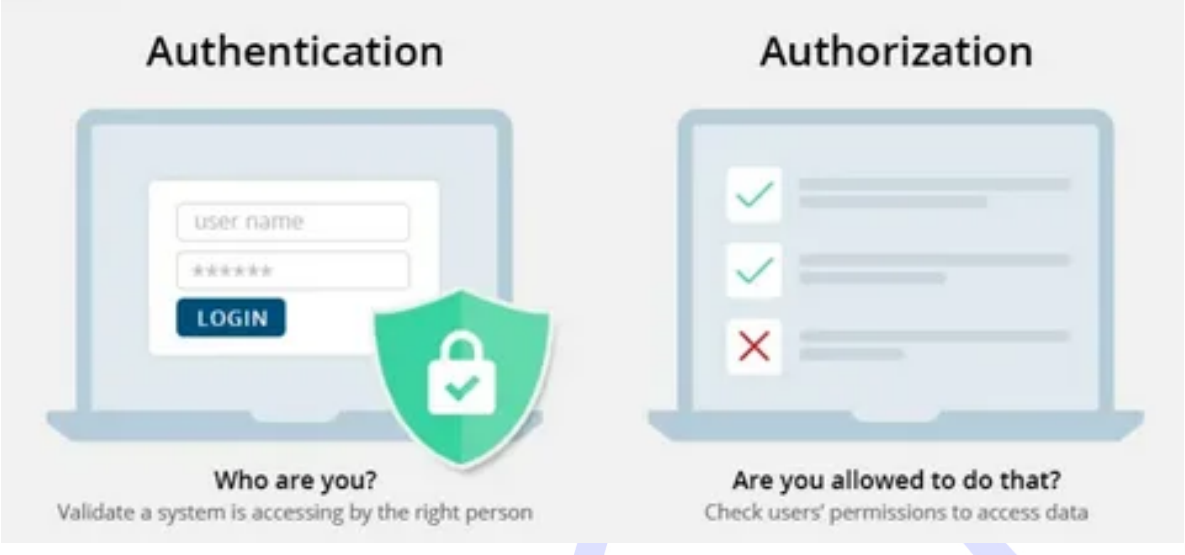
Зачем нужен DRF

- CRUD

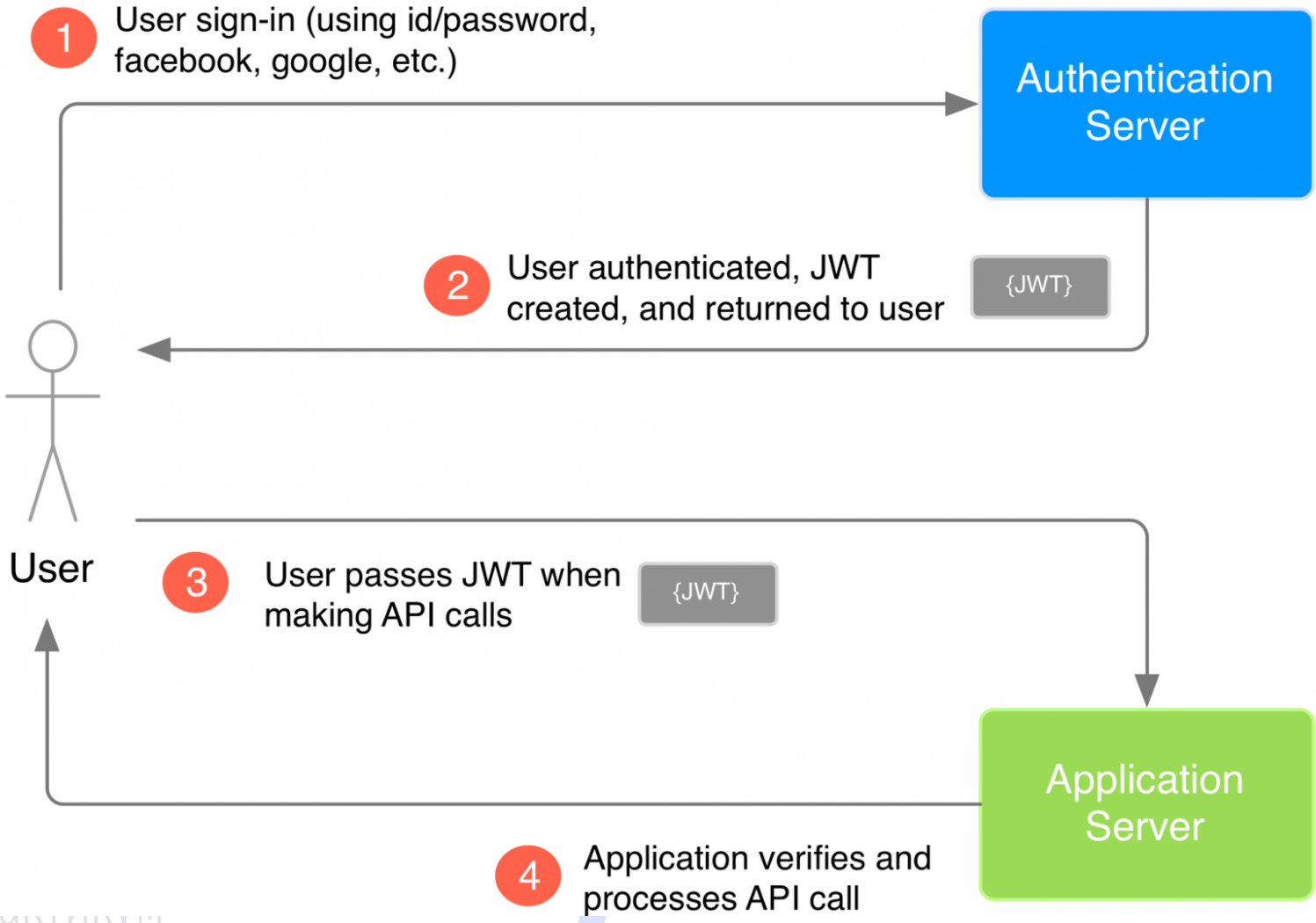
REST	HTTP	
Create	Post	POST https://api.twitter.com/1.1/statuses/retweet/241259202004267009.json
Read	Get	GET https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=twitterapi &count=2
Update	Put	PUT https://www.googleapis.com/calendar/v3/calendars/calendarId/events/eventId
Delete	Delete	DELETE https://www.googleapis.com/calendar/v3/calendars/*calendarId*/events/*eventId*

Зачем нужен DRF

- Права доступа



JWT



Зачем нужен DRF

- Сериализация

```
import json
from django.db import Model


class MyModel(Model):
    ...
    def to_json(self):
        ...
```

Зачем тогда нужны сериализаторы?

Serializers

- JSON
- XML
- YAML
- многие другие

Serializers. Валидация

```
def validate_<field>(self, field):
    if ...:
        raise serializers.ValidationError
    return field

def validate(self, items):
    return items
```

Serializers. Default

```
def perform_create(self, serializer):
    serializer.save(
        owner=self.request.user
    )
```

Это плохой стиль! Необходимо использовать:

- CurrentUserDefault
- CreateOnlyDefault

ViewSet

```
class ModelViewSet(
        mixins.CreateModelMixin,
        mixins.RetrieveModelMixin,
        mixins.UpdateModelMixin,
        mixins.DestroyModelMixin,
        mixins.ListModelMixin,
        GenericViewSet
):
```

ViewSet. Ограничение на методы

1. Можно наследоваться не от *ModelViewSet* а создать свой набор миксинов

```
class SnippetsViewSet(
        mixins.CreateModelMixin,
        mixins.RetrieveModelMixin,
        mixins.ListModelMixin
):
```

1. Можно использовать *http_method_names*

```
class SnippetsViewSet(ModelViewSet):
    ...
    http_method_names = ["get", "post"]
```

ViewSet. create / performe_create

```python
class CreateModelMixin:
    """
    Create a model instance.
    """
    def create(self, request, *a, **k):
        serializer = self.get_serializer(
          data=request.data)
        serializer.is_valid(
          raise_exception=True)
        self.perform_create(serializer)
        # ...

    def perform_create(self, serializer):
        serializer.save()
```

ViewSet. get_serializer_class

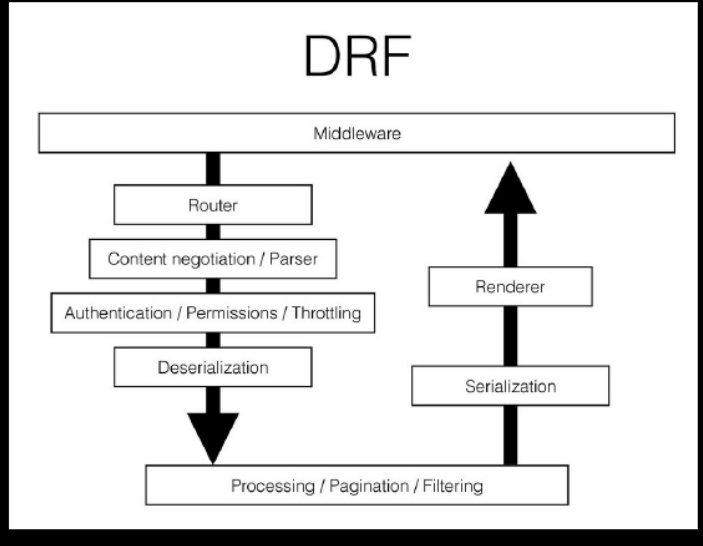
```
def get_serializer(self, *args, **kwargs):
    """
    Return the serializer instance that should be used for validating and
    deserializing input, and for serializing output.
    """
    serializer_class = self.get_serializer_class()
    kwargs.setdefault('context', self.get_serializer_context())
    return serializer_class(*args, **kwargs)

def get_serializer_class(self):
    """
    Return the class to use for the serializer.
    Defaults to using `self.serializer_class`.

    You may want to override this if you need to provide different
    serializations depending on the incoming request.

    (Eg. admins get full serialization, others get basic serialization)
    """
    assert self.serializer_class is not None, (
        "'%s' should either include a `serializer_class` attribute, "
        "or override the `get_serializer_class()` method."
        % self.__class__.__name__
    )

    return self.serializer_class
```

Permissions

DRF. Permissions

1) Global/project

```
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
        # 'rest_framework.permissions.AllowAny',
        # 'rest_framework.permissions.IsAuthenticatedOrReadOnly',
        # 'rest_framework.permissions.IsAdminUser',
    ]
}
```



Permissions

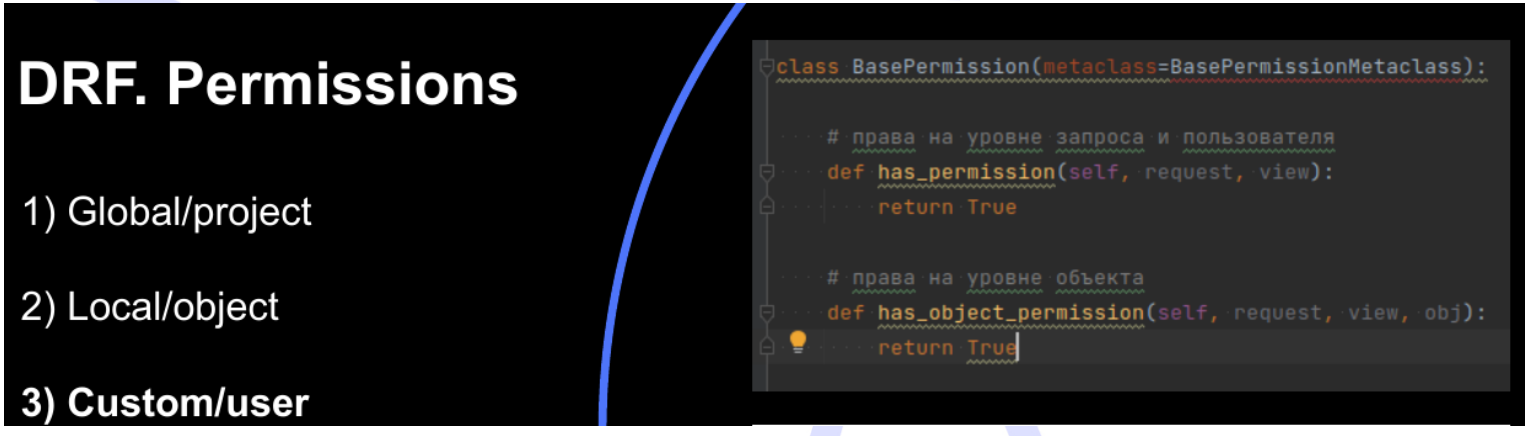
DRF. Permissions

1) Global/project

2) **Local/object**

```python
class CustomerViewSet(viewsets.ModelViewSet):
    queryset = Customer.objects.all()
    serializer_class = CustomerSerializer
    # Устанавливаем разрешение
    permission_classes = (permissions.IsAuthenticatedOrReadOnly,)
```

Permissions



- has_permission
- has_object_permission

Паджинация

DRF. Pagination

1) Global/project

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.LimitOffsetPagination',
    'PAGE_SIZE': 100
}
```

2) Local/custom

```
class StandardResultsSetPagination(PageNumberPagination):
    page_size = 100
    page_size_query_param = 'page_size'
    max_page_size = 1000

class BillingRecordsView(generics.ListAPIView):
    queryset = Billing.objects.all()
    serializer_class = BillingRecordsSerializer
    pagination_class = StandardResultsSetPagination
```

```
HTTP 200 OK
{
    "count": 1023
    "next": "https://api.example.org/accounts/?page=5",
    "previous": "https://api.example.org/accounts/?page=3",
    "results": [
       …
    ]
}
```

Throttling

DRF. Throttling

1) Global/project

```
REST_FRAMEWORK = {
    'DEFAULT_THROTTLE_CLASSES': [
        'rest_framework.throttling.AnonRateThrottle',
        'rest_framework.throttling.UserRateThrottle'
    ],
    'DEFAULT_THROTTLE_RATES': {
        'anon': '100/day',
        'user': '1000/day'
    }
}
```

2) Local/custom

```
class ProductList(generics.ListAPIView):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
    permission_classes = (OwnerOrReadOnly,)
    throttle_classes = (AnonRateThrottle,)
```

- При превышении лимита
  будет возвращен код ответа “429 Too many Requests”

Ответ будет содержать сообщение об ошибке
и время ограничения

```
HTTP/1.1 429 Too Many Requests
Content-Type: text/html
Retry-After: 3600
```

Фильтрация

- Django-Filter иногда бывает слишком громоздким решением для простой задачи

```
def get_queryset(self):
    langs = (
        self.request.query_params
        .getlist('langs')
    )
    qs = Recipe.objects
    if tags:
        qs = qs.filter(language__in=langs)
    # if self.request.query_params.get(
    #         'is_favorited'):
    #     qs = qs.filter(is_favorited=True)
    return qs
```

Дополнительная литература

- JWT токен
- DRF Tutorial