Maxim Danilov
Developer in winePad

RESTful-API

with raw Django

Who I am

- Professional skier 🎿
- Trail runner 🏃
- Sommelier 🍷🍾
- Mentor for developer community for free 🤓

Special thanks to my family for the support

My wife
Elena

daughter
Maja

son
Mark

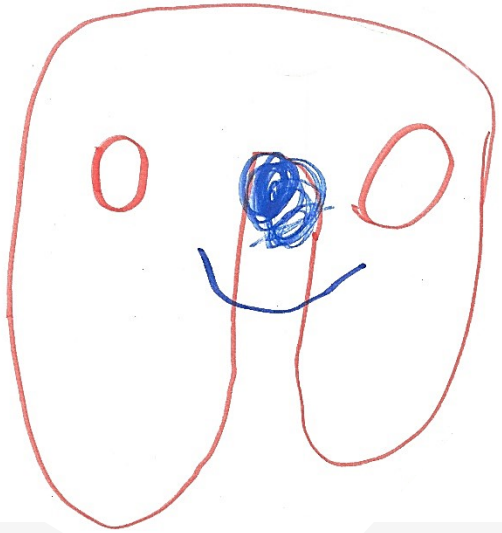
cat
Kisa

dog
Marselle

Thanks to Mark & Maja for the illustrations

Links

- https://github.com/danilovmy/rawDjangoAPI
- Slides & code samples from this workshop



What is REST (Re-presentative State Transfer)

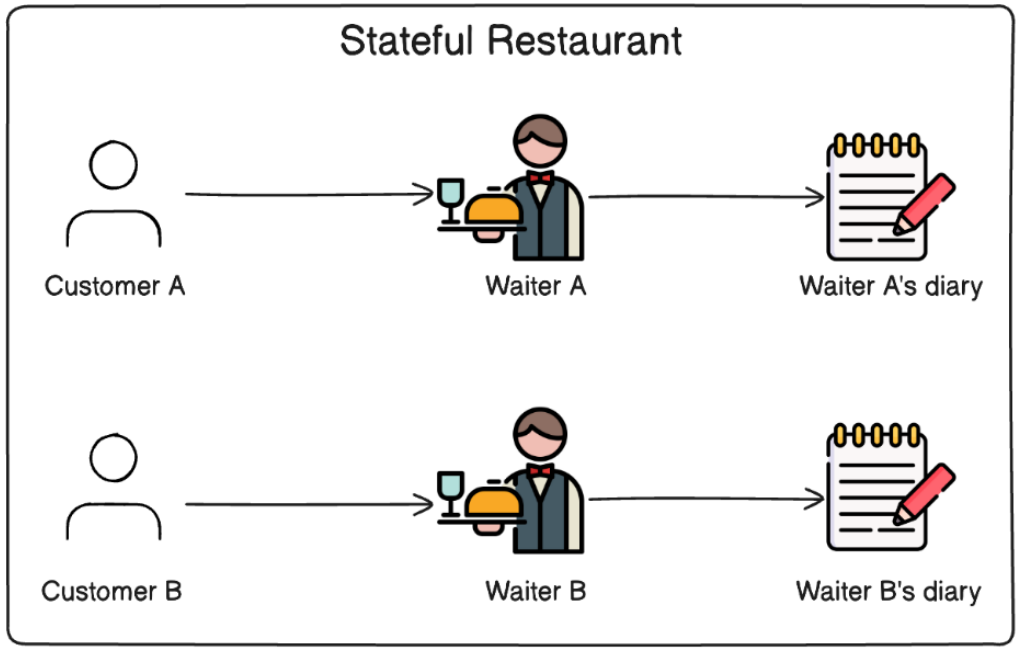
- 6 Principles, how server application can be built
- Author Roy Fielding, 2000

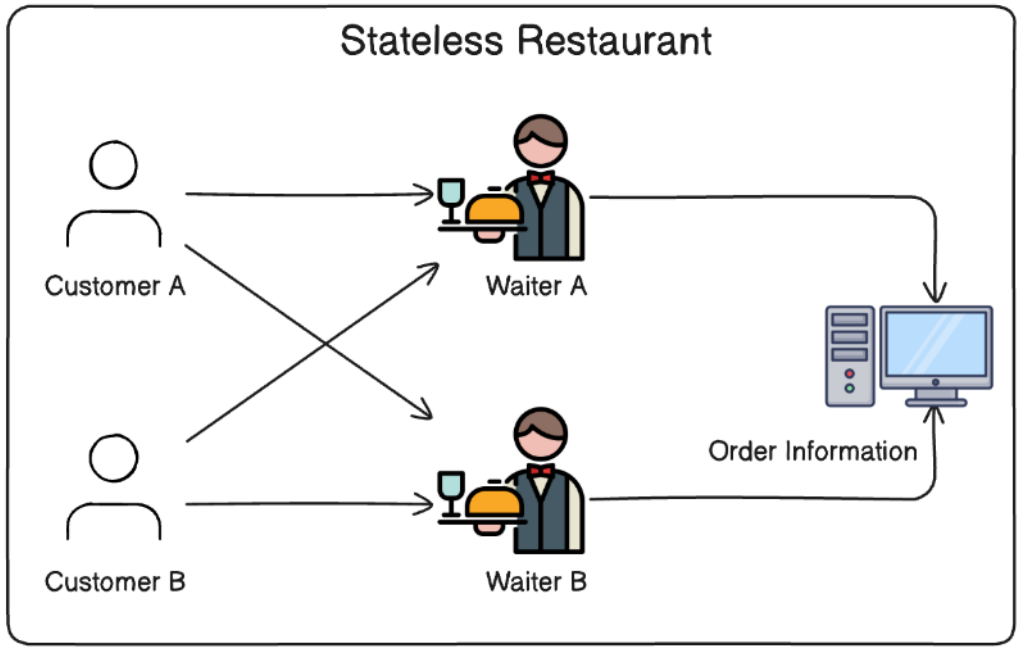
REST priciples (1 from 6)

- Server independent from clients

REST priciples (2 from 6)

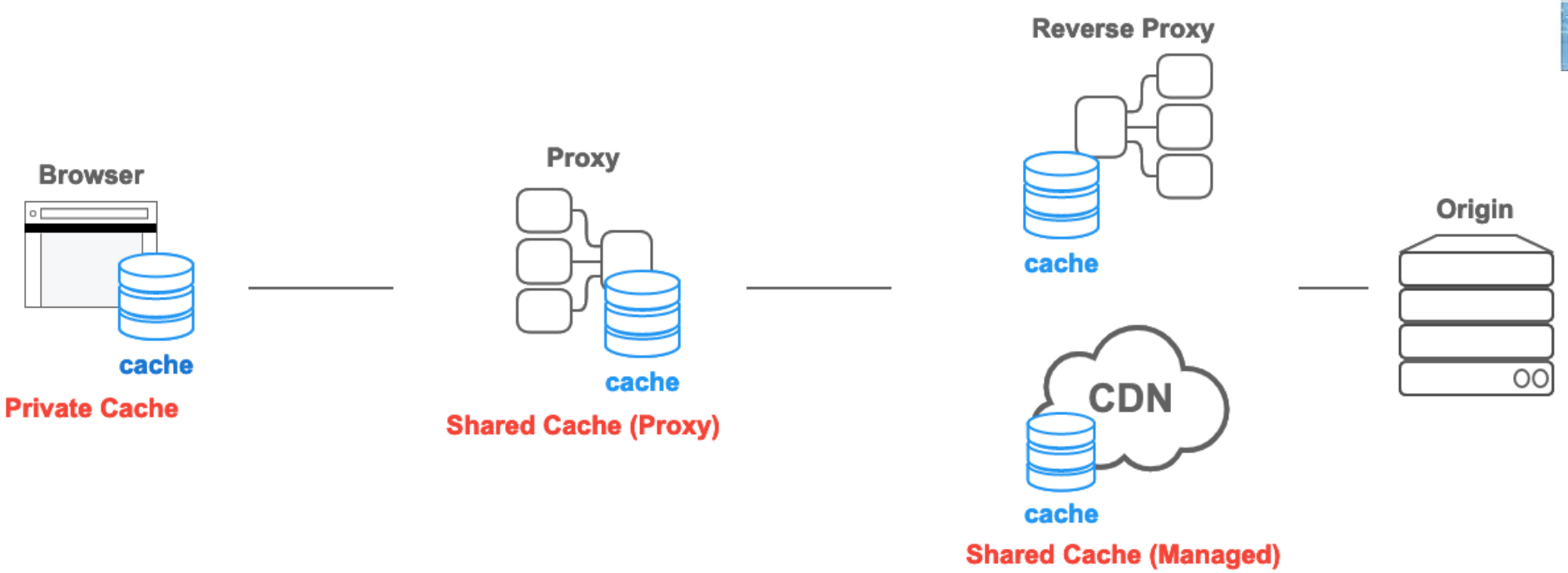
- Statelessness





REST priciples (3 from 6)

- Cacheability

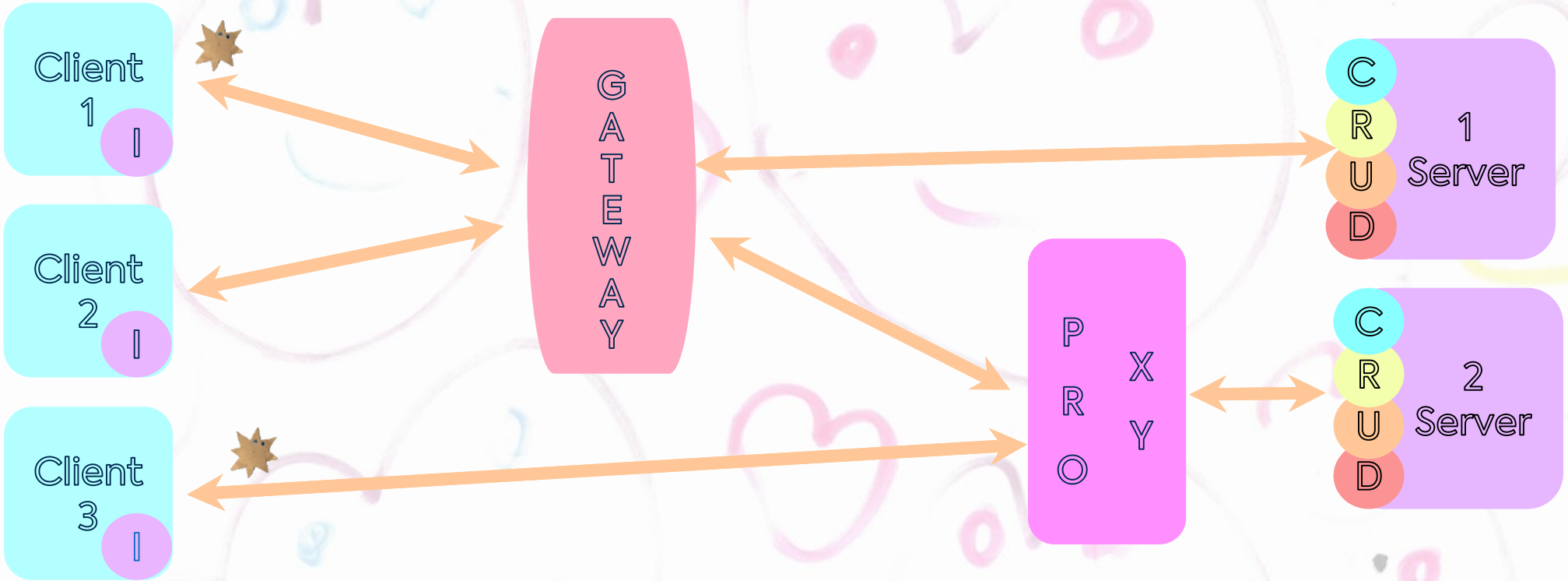


REST priciples (4 from 6)

- Uniform interface

REST priciples (5 from 6)

- Layered server system



REST priciples (6 from 6)

- Optional: code-on-demand

7.1, RFC 4329. obsolete
GET /product/id HTTP/1.1
Content-Type: content-type: text/javascript; charset=utf-8

7.2, RFC 4329
GET /product/id HTTP/1.1
Content-Type: content-type: application/javascript; charset=utf-8

REST priciples (>6 from 6)

- Any other suggestions are just a extension of Roy Fielding's original idea

A good example of REST documentation from AWS
https://aws.amazon.com/what-is/restful-api/

RESTful api

- APIs following the REST architectural style are REST APIs
- Web services containing REST APIs are RESTful web services.
- Under RESTful APIs are meant RESTful web services.

APIs in Django are not RESTful... ☹

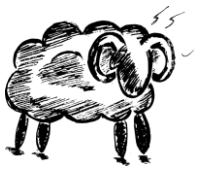
- Generally, stateful (due sessions) (❌ RP1)

- The Routing for Request Methods sepately not in a box (❌ RP4)
- Handlers for TRACE, HEADS, PATCH not in the box (❌ RP4)

- API autodocumentation is not in the box

Why not other frameworks?

- Flask
- FastApi
- Robin
- Litestar(Starlite)
- Blacksheep
- ...



Frame for work

- Django is smaller than many other but “all in box”

Frame for work

- Django is smaller than many other but “all in box”

Frame for work

- Django is smaller than many other but “all in box”

- Allowed One file project (μ-Django)

Frame for work

- Django is smaller than many other but “all in box”

- Allowed One file project (μ-Django)

Frame for work

- Django is smaller than many other but “all in box”
- Allowed One file project (μ-Django)
- Django offers standard “frame for work”

Frame for work

- Django is smaller than many other but “all in box”
- Allowed One file project (μ-Django)
- Django offers standard “frame for work”

Code generators:

https://tools.openapis.org/

openapi-generator-cli

- popular OpenAPI tool
- supported Python 3.14
- complex, verbose

openapi-python-generator & Co

- Flask-FastAPI-oriented
- clean final results
- small, few settings

Generate models:

openapi-generator-cli

schema.yaml -> sql -> db -> manage.py inspectdb

Generate models:

openapi-generator-cli

schema.yaml -> sql -> db -> manage.py inspectdb

Generate views:

openapi-generator-cli

schema.yaml -> generate views.py

Generate views:

openapi-generator-cli

schema.yaml -> generate views.py

Routes (Django style):

The GET, POST, PATCH... requests to the same URL are handled by the same view.

Routes (Django style):

The GET, POST, PATCH... requests to the same URL are handled by the same view.

Django-GCBV views:

request.GET -> ListView
request.POST -> CreateView (through ListView)

request.GET + .../obj_id/ -> DetailView
request.PUT (PATCH) + .../obj_id/ -> UpdateView (through DetailView)
request.DELETE + .../obj_id/ -> DeleteView (through DetailView)

Django-GCBV views:

request.GET -> ListView
request.POST -> CreateView (through ListView)

request.GET + .../obj_id/ -> DetailView
request.PUT (PATCH) + .../obj_id/ -> UpdateView (through DetailView)
request.DELETE + .../obj_id/ -> DeleteView (through DetailView)

APIs elements in Django:

webserver

middleware

url-dispatcher

view.as_view

Validators

Models ORM

Collect context

serializers

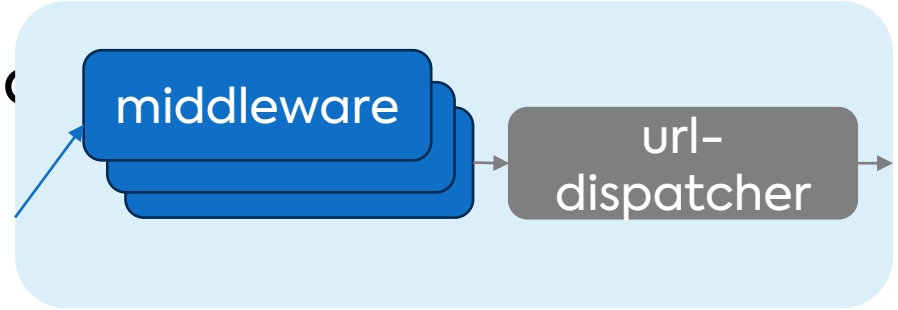
render to response

middleware

Django

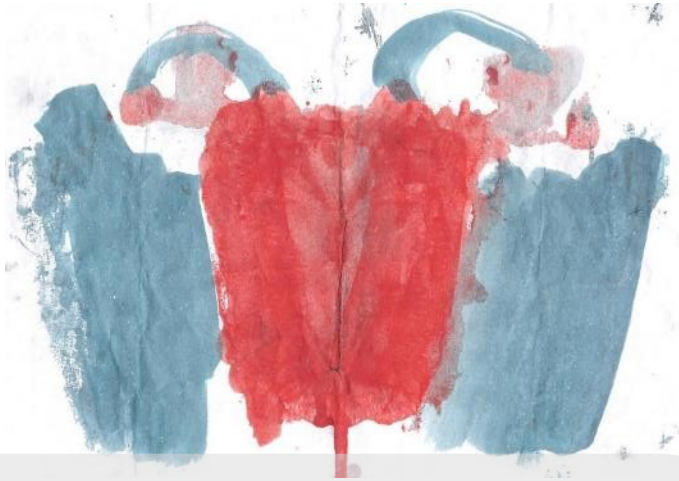
APIs elements in Django:

- Middleware in Django



- Applied on every request
- Reduced performance
- Breaks statelessy
- Not every middleware suited for every request

Middlware alternatives:



- method / function decorator

- from django.utils.decorators import method_decorator
- from django.utils.decorators import decorator_from_middleware_with_args
- cache_func = decorator_from_middleware_with_args(CacheMiddleware)
- @method_decorator('get', cache_func(3600))
- def ProductDetail(DetailView):
- ...

What's new in Django 5.2 o 5.2

Decorators ¶

- `method_decorator()` now supports wrapping asynchronous view methods.

- @method_decorator('get', cache_func(3600))
- def ProductDetail(DetailView):
- ...
- async def get(self, *args, **kwargs):
- ...

APIs elements in Django:

- Url dispatcher in Django:

- Class URLResolver aka “urlpatterns”
- Provide appropriated handler by URI from request
- Single url or group of urls can be wrapped by decorator instead of using middleware.

Middlware alternatives:

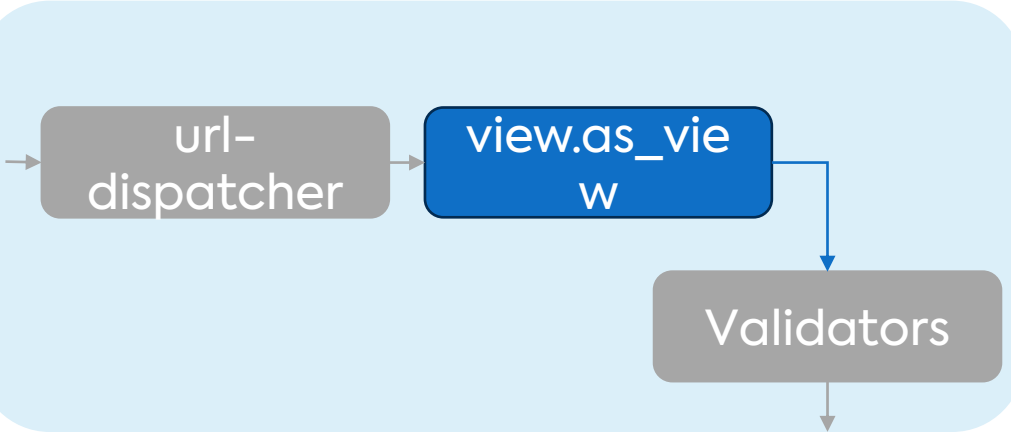
- wrappers for groups of urls (still not async)

- from authlib.integrations.django_oauth2 import ResourceProtector

- jwt_protector = ResourceProtector()
- urlpatterns = [path('api/', jwt_protector(include('api.urls')))]

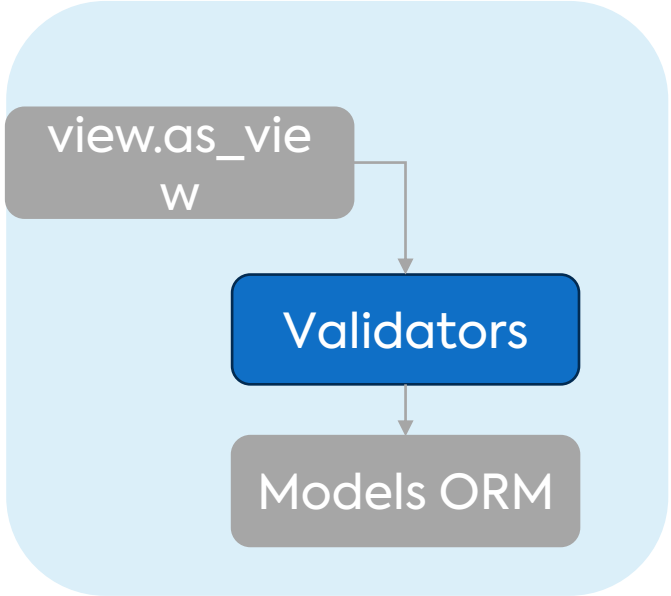
APIs elements in Django:

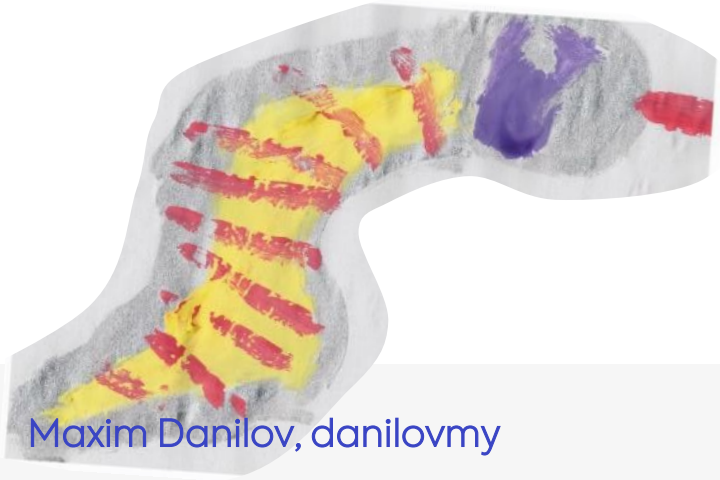
- sync/async FB views or GCB views
- GCBView.dispatch Provide handler
- for appropriate request.method
- response object should be returned



APIs elements in Django:

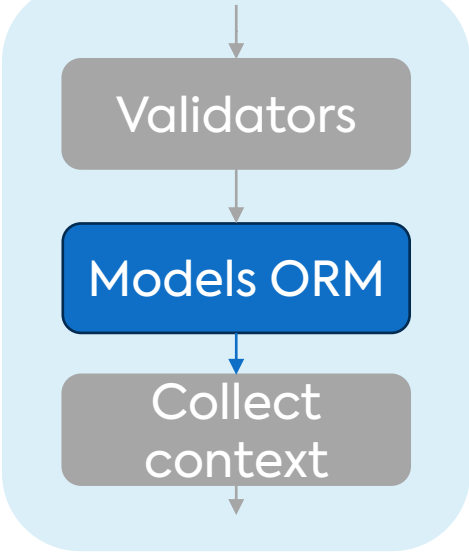
- Django.forms.Form as a data validator.
- Simple data validation with Validators
- Complex validation on Form._post_clean()





APIs elements in Django:

- Model. refresh_from_db() – to fetch data from DB
- Model.save() – to save data to DB
- Complex DB validation on Model.full_clean()
  - Model.clean()
  - Model.validate_unique()
- Some methods formally a-sync



What's new in Django 5

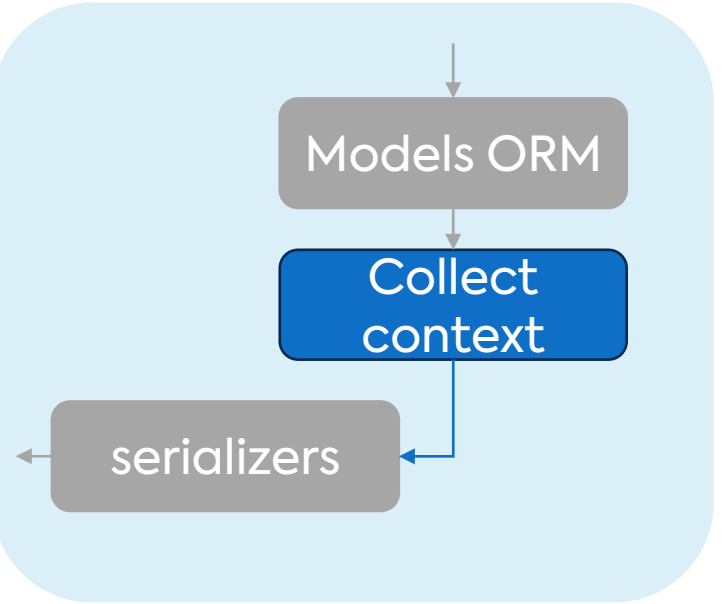
Signals ¶

- The new **`Signal.asend()`** and **`Signal.asend_robust()`** methods allow asynchronous signal dispatch. Signal receivers may be synchronous or asynchronous, and will be automatically adapted to the correct calling style.

- @receiver(my_signal)
- async def my_async_receiver(sender, **kwargs):
- await logging.ainfo(f"{sender} signal received")

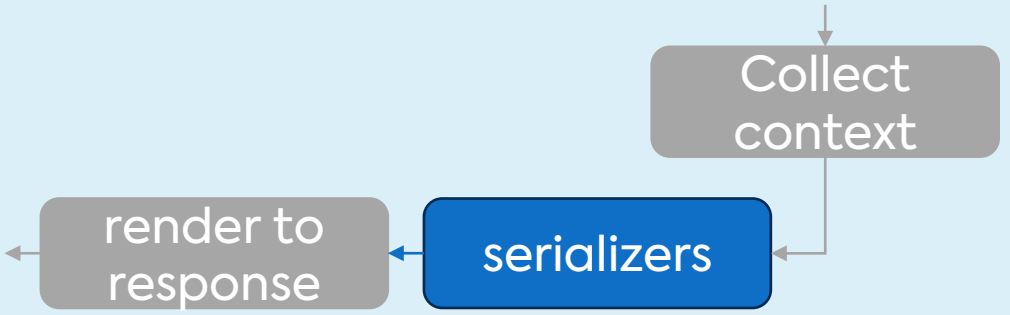
APIs elements in Django:

- Weakest part in GCBV, not designed to be async
- Designed to perform final business logic



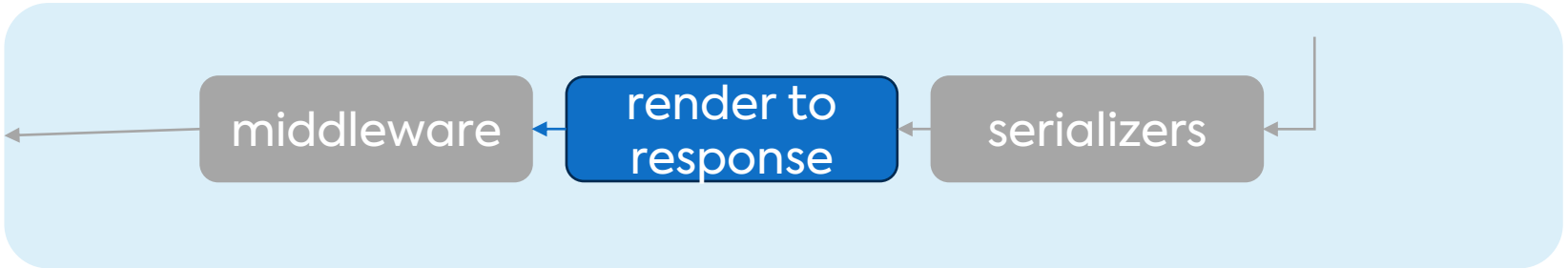
APIs elements in Django:

- Django.core.serializer as a result serializer, (improvements in Dj5.2)
- Knows, how to serialize specific data.
- Should not perform data validation
- Not designed to be async!



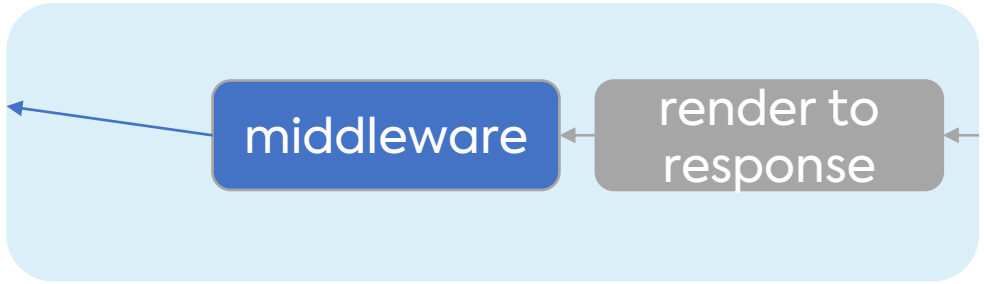
APIs elements in Django:

- JsonResponse / HttpResponse class for response.
- StreamingHttpResponse for partially prepeared data (async)

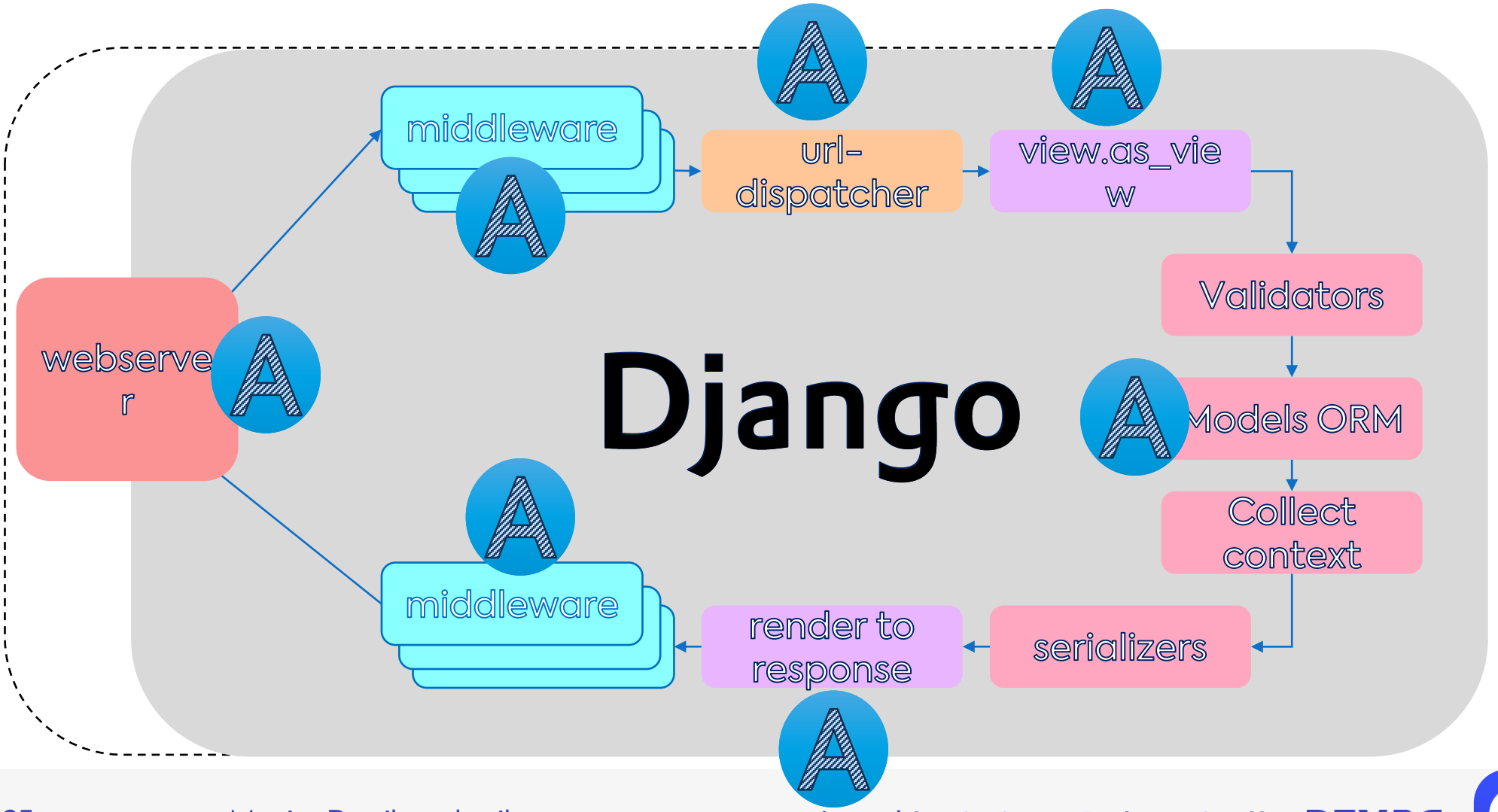


APIs elements in Django:

- Last middleware in Django „convert_exception_to_response”.
- To serialize Exception from Django.core.exceptions in Response.



Async APIs elements in Django:



What's new in Django 5

- Auth/permissions decorator (in general async)
  - The new asynchronous functions are now provided, using an **a** prefix: **`django.contrib.auth.aauthenticate()`**, **`aget_user()`**, **`alogin()`**, **`alogout()`**, and **`aupdate_session_auth_hash()`**.
  - **`AuthenticationMiddleware`** now adds an **`HttpRequest.auser()`** asynchronous method that returns the currently logged-in user.
- Cache decorators (async in last version)

Decorators ¶

- The following decorators now support wrapping asynchronous view functions:
  - **`cache_control()`**

REST API Autodocumentation:

- Request.method.OPTION to provide yaml documentation about the API
- Handler.doctring used to provide information for yaml generation
- dockstring is updated automatically by the linter

- www.youtube.com/watch?v=MeoKMdyPRWo

Summary:

- Raw Django has EVERYTHING you need to create REST APIs
- REST APIs in Raw Django can be a synchronous or asynchronous
- Raw Django is not ready to offer full async functionality
- Know you framework for fun and profit!

Thank you for your attention!

- Maxim Danilov
- maxim@wpshop.at

Link to slides & code

https:/bitbucket.org/danilovmy/pyconlt25