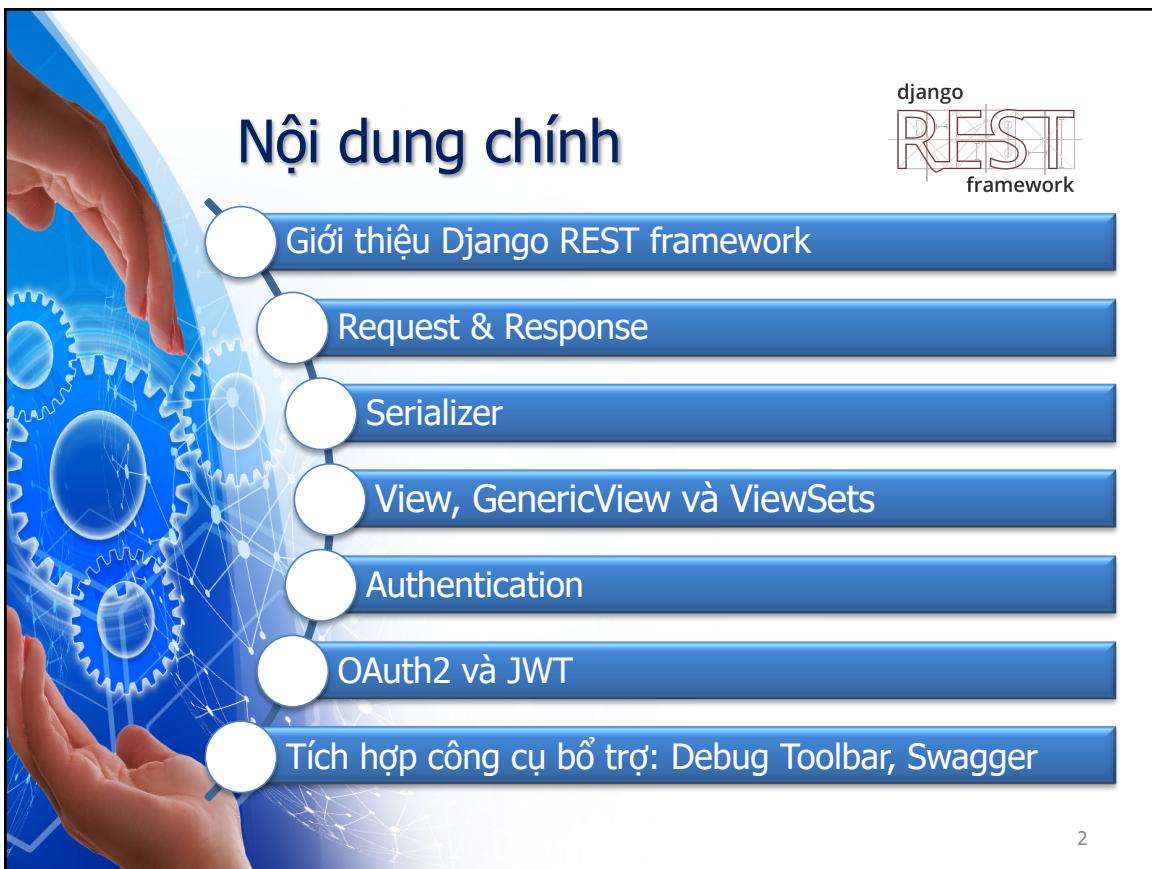




1



2



Giới thiệu Django Rest API

- Django REST framework (DRF) là bộ công cụ mạnh và linh hoạt để phát triển API.
- Một số ưu điểm sử dụng Django REST API.
 - Hỗ trợ giao diện duyệt API hiệu quả cho dev.
 - Hỗ trợ nhiều authentication policies, chứng thực bao gồm OAuth1a và OAuth2.
 - Serialization hỗ trợ cả các data source theo ORM và không ORM (non-ORM).
 - Tài liệu phong phú và cộng đồng lớn.

Dương Hữu Thành

3

3



Cài đặt Django Rest API

- Cài đặt môi trường
 - pip install django
 - pip install djangorestframework**
 - pip install markdown
 - pip install django-filter
- Cập nhật biến INSTALLED_APPS

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

Dương Hữu Thành

4

4



Viết API đầu tiên

- Tạo courses/serializers.py

```
from rest_framework.serializers import ModelSerializer
from .models import Course

class CourseSerializer(ModelSerializer):
    class Meta:
        model = Course
        fields = ['id', 'subject', 'created_date', 'category']
```

Dương Hữu Thành

5



Viết API đầu tiên

- Viết courses/views.py

```
from rest_framework import viewsets, permissions
from .models import Course
from .serializers import CourseSerializer

class CourseViewSet(viewsets.ModelViewSet):
    queryset = Course.objects.filter(active=True)
    serializer_class = CourseSerializer
    permission_classes = [permissions.IsAuthenticated]
```

Dương Hữu Thành

6

Viết API đầu tiên

- Viết courses/urls.py

```
from django.urls import path, include
from . import views
from rest_framework import routers

router = routers.DefaultRouter()
router.register('courses', views.CourseViewSet)

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

- Chạy server và truy cập http://127.0.0.1:8000/

Dương Hữu Thành

7

Api Root

The default basic root view for DefaultRouter

GET /

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
 "courses": "http://127.0.0.1:8000/courses/"
}

Dương Hữu Thành

8

8



Viết API đầu tiên

← → ⌛ 127.0.0.1:8000/courses/ ☆ 🚙 📁 ⋮

Django REST framework admin

Api Root / Course List

Course List

OPTIONS GET

GET /courses/

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[{"id": 1, "subject": "Các công nghệ lập trình hiện đại", "created_date": "2021-05-29T01:35:34.495737Z", "category_id": 1}]
```

Raw data HTML form

Subject POST

Dương Hữu Thành 9



Viết API đầu tiên

Course List – Django REST framework + ⌛ 127.0.0.1:8000/courses/ ☆ 🚙 📁 ⋮

Django REST framework

Api Root / Course List

Course List

GET

GET /courses/

```
HTTP 403 Forbidden
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Authentication credentials were not provided."
}
```

Dương Hữu Thành 10

Viết API đầu tiên

- Phân trang cho phép chỉ định số phần tử hiển thị trên từng trang.
- Truy cập vào admin thêm vài khóa học

Django administration

Home : Courses > Courses

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

COURSES

Categories + Add

Courses + Add

Lessons + Add

Select course to change

Action: — Go 0 of 8 selected

COURSE

Cơ sở lập trình

Kỹ thuật lập trình

Cấu trúc dữ liệu và giải thuật

Công nghệ phần mềm

Kiểm thử phần mềm

Thiết kế Web

Lập trình Java

Các công nghệ lập trình hiện đại

8 courses

ADD COURSE +

Dương Hữu Thành

11

11

Viết API đầu tiên

- Ta phân mảng trang hiển thị tối đa 2 khóa học.
- Trong tập tin settings.py thiết lập cấu hình sau

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS':
        'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 2
}
```

- Truy cập <http://localhost:8000/courses/>

Viết API đầu tiên

Course List

OPTIONS GET

Click vào đây xem các trang hiện thị khóa học

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 8,
    "next": "http://localhost:8000/courses/?page=2",
    "previous": null,
    "results": [
        {
            "id": 1,
            "subject": "Các công nghệ lập trình hiện đại",
            "created_date": "2021-05-29T01:35:34.495737Z",
            "category_id": 1
        },
        {
            "id": 2,
            "subject": "Lập trình Java",
            "created_date": "2021-06-01T03:09:56.146219Z",
            "category_id": 1
        }
    ]
}
```

Dương Hữu Thành

GET /courses/?page=3

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 8,
    "next": "http://localhost:8000/courses/?page=4",
    "previous": "http://localhost:8000/courses/?page=2",
    "results": [
        {
            "id": 5,
            "subject": "Công nghệ phần mềm",
            "created_date": "2021-06-01T03:11:48.207315Z",
            "category_id": 1
        },
        {
            "id": 6,
            "subject": "Cấu trúc dữ liệu và giải thuật",
            "created_date": "2021-06-01T03:12:48.412100Z",
            "category_id": 1
        }
    ]
}
```

13

Request

- Request của REST framework **kế thừa từ HttpRequest chuẩn**, nhưng linh hoạt hơn trong xử lý và chứng thực request (request parsing và request authentication).
- Request giúp cho việc xử lý dữ liệu request dạng json hay một media type nào khác theo **cùng cách thức** làm việc với dữ liệu form.

Dương Hữu Thành

14

14



Các thuộc tính của Request

- `request.data`: dữ liệu của request body.
- `request.query_params`: các tham số truyền từ request get.
- `request.method`: trả về phương thức (chữ in hoa) thực hiện request như GET, POST, PUT, DELETE, PATCH.
- `request.content_type`: trả về chuỗi đại diện cho media type của request body.



Các thuộc tính của Request

- `request.user`: các user đã chứng thực thì nó trả về thể hiện của
 - `django.contrib.auth.models.User`
- Ngược lại, nó là thể hiện của
 - `django.contrib.auth.models.AnonymousUser`
- `request.auth`: chứa thông tin bổ sung chứng thực, nó phụ thuộc cách thức chứng thực, nhưng nó thường không chứa thông tin token của request.



Các thuộc tính của Request

- **request.authenticators**: các chính sách chứng thực, trong các lớp APIView hoặc @api_view thuộc tính được thiết lập thông qua thuộc tính authentication_classes hoặc giá trị thuộc tính cấu hình DEFAULT_AUTHENTICATORS.

Dương Hữu Thành

17

17



Response

- REST framework cung cấp lớp Response trả về nội dung với nhiều dạng media type phụ thuộc client request. Lớp Response là lớp con của Django SimpleTemplateResponse.
- Cú pháp tạo response:

```
Response(data, status=None, template_name=None,  
         headers=None, content_type=None)
```

Dương Hữu Thành

18

18



Các thuộc tính của Response

- **data**: dữ liệu đã được serialize.
- **status**: status code cho response, mặc định 200.
- **headers**: từ điển HTTP headers cho response.
- **content_type**: content type cho response.
- **template_name**: tên template sử dụng nếu HTMLRenderer được chọn.



Serializer

- **Serializer** cho phép **chuyển các dữ liệu phức tạp** như querysets hoặc thể hiện của model thành các kiểu dữ liệu python và dễ dàng **kết xuất** thành dữ liệu **JSON**, XML hay một content type nào khác.
- Nó cũng cung cấp khả năng **deserializer** để **chuyển** dữ liệu đã được **chuyển** về **kiểu dữ liệu** phức tạp ban đầu sau khi đã kiểm tra (validate).



Serializer

- Phương thức **validate()** dùng để kiểm tra chung cho nhiều trường trong serializer, phương thức có một đối số duy nhất là từ điển chứa giá trị các trường.
- Phương thức **validate_<field_name>** sẽ trả về giá trị kiểm tra của trường tương ứng hoặc ném ngoại lệ serializers.ValidationError.
- Chú ý nếu <field_name> được khai báo required là False thì validate sẽ không được thực hiện.

Dương Hữu Thành

21

21



Serializer

- **Lớp Serializer** cung cấp các phương thức chung để xử lý output cho response.
- **Lớp ModelSerializer** giúp nhanh chóng tạo lớp Serializer với **các trường tương ứng với các trường của Model**. Khi đó
 - Tự tạo các field giống model.
 - Tự tạo validator cho serializer.
 - Bao gồm các phương thức mặc định như save, update, create.

Dương Hữu Thành

22

22



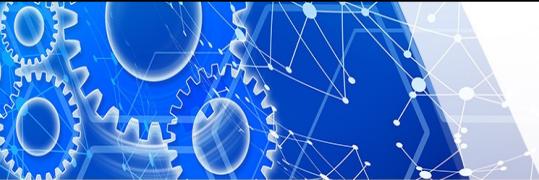
Serializer

- Lớp `HyperlinkedModelSerializer` tương tự `ModelSerializer`, nhưng nó sử dụng siêu liên kết đại diện cho các quan hệ.

Dương Hữu Thành

23

23



ModelSerializer

- Thuộc tính trong meta class
 - `model`: lớp model liên kết cho serialize.
 - `fields`: chỉ định các field sẽ được serialize, thiết lập là '`__all__`' để serializer tất cả field.
 - `read_only_fields`: chỉ định các field chỉ đọc.
 - `exclude`: chỉ định các field không serialize.
 - `extra_kwargs`: bổ sung một số thông tin ràng buộc trên fields.
 - `deep`

Dương Hữu Thành

24

24

ModelSerializer

- Ví dụ tạo serializer cho Lesson

```
class TagSerializer(ModelSerializer):  
    class Meta:  
        model = Tag  
        fields = ['id', 'name']  
  
class LessonSerializer(ModelSerializer):  
    tags = TagSerializer(many=True)  
  
    class Meta:  
        model = Lesson  
        fields = ['id', 'subject', 'content',  
                  'created_date', 'updated_date', 'tags']
```

Dương Hữu Thành

25

25

ModelSerializer

- Sử dụng cho 1 đối tượng QuerySet

```
lesson = Lesson.objects.get(pk=1)  
LessonSerializer(lesson).data
```

- Sử dụng cho danh sách QuerySet

```
lessons = Lesson.objects.filter(active=True)  
LessonSerializer(lessons, many=True).data
```

- Cung cấp thêm thông tin context cho serializer

```
LessonSerializer(lesson,  
                 context={'request': request}).data
```

Dương Hữu Thành

26

26

The screenshot shows a browser window titled "Lesson Instance – Django REST". The URL is "localhost:8000/lessons/1/". The page title is "Django REST framework". The navigation bar includes "Api Root / Lesson List / Lesson Instance". Below the navigation is a section titled "Lesson Instance" with "OPTIONS" and "GET" buttons. A "GET /lessons/1/" button is also present. The main content area displays a JSON response:

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "subject": "Giới thiệu",
    "content": "Giới thiệu các công nghệ lập trình hiện đại",
    "created_date": "2021-05-30T01:59:15.931582Z",
    "updated_date": "2021-05-30T07:41:17.071208Z",
    "tags": [
        {
            "id": 1,
            "name": "nhập môn"
        }
    ]
}
```

Dương Hữu Thành

27

The slide has a blue gear and network background. The title is "ModelSerializer". The content is a bulleted list:

- Sau đây 2 serializer minh họa việc chỉnh sửa hoặc ghi đè xử lý của serializer
 - Ví dụ 1: chỉnh sửa thông tin serialize image của CourseSerializer trả về đường dẫn tuyệt đối cho client.
 - Ví dụ 2: ghi đè phương thức create của UserSerializer trong chức năng đăng ký người dùng để băm mật khẩu trước khi lưu xuống CSDL.

Dương Hữu Thành

28



ModelSerializer

```
class CourseSerializer(HyperlinkedModelSerializer):  
    image = serializers.SerializerMethodField(source='image')  
  
    def get_image(self, obj):  
        request = self.context['request']  
        if obj.image.name.startswith('static/'):  
            path = "/%s" % obj.image.name  
        else:  
            path = '/static/%s' % (obj.image)  
  
        return request.build_absolute_uri(path)  
  
    class Meta:  
        model = Course  
        fields = ['id', 'subject', 'image',  
                  'created_date', 'category_id']
```

Dương Hữu Thành

25

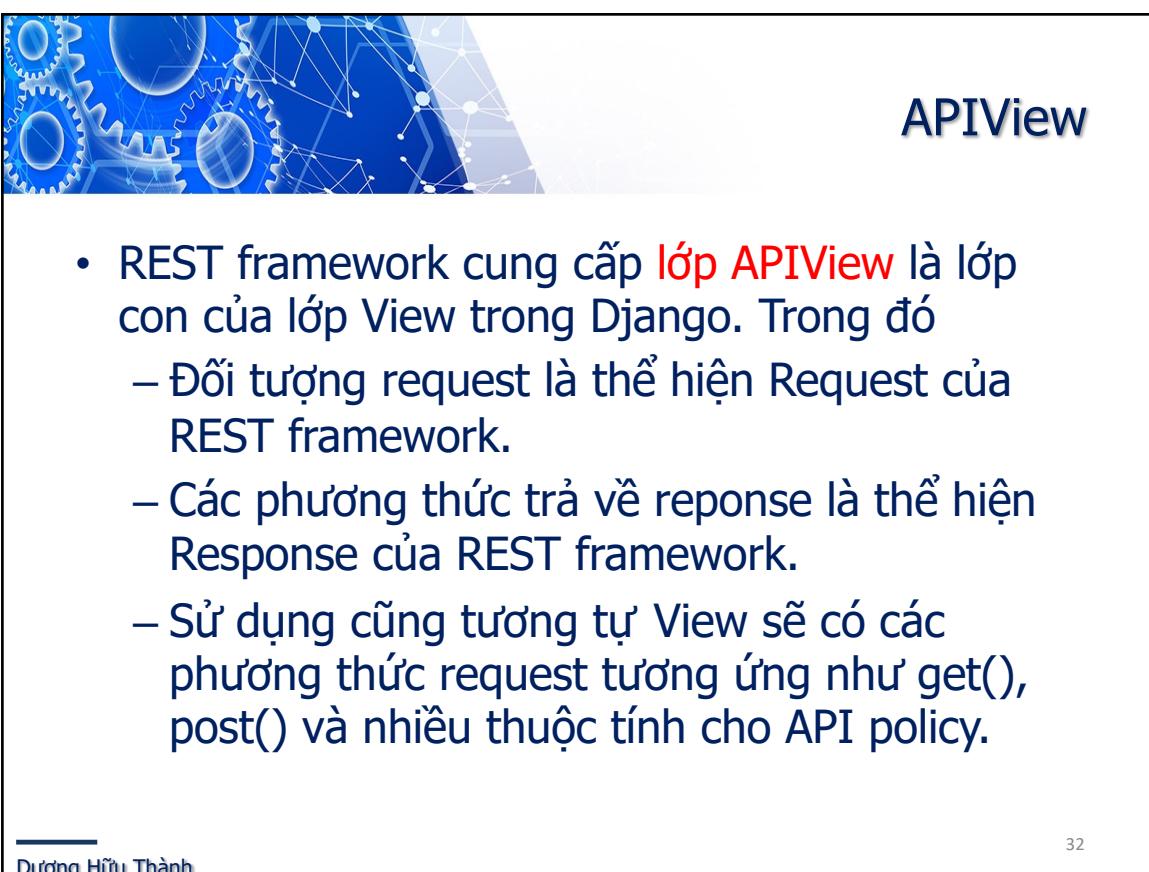
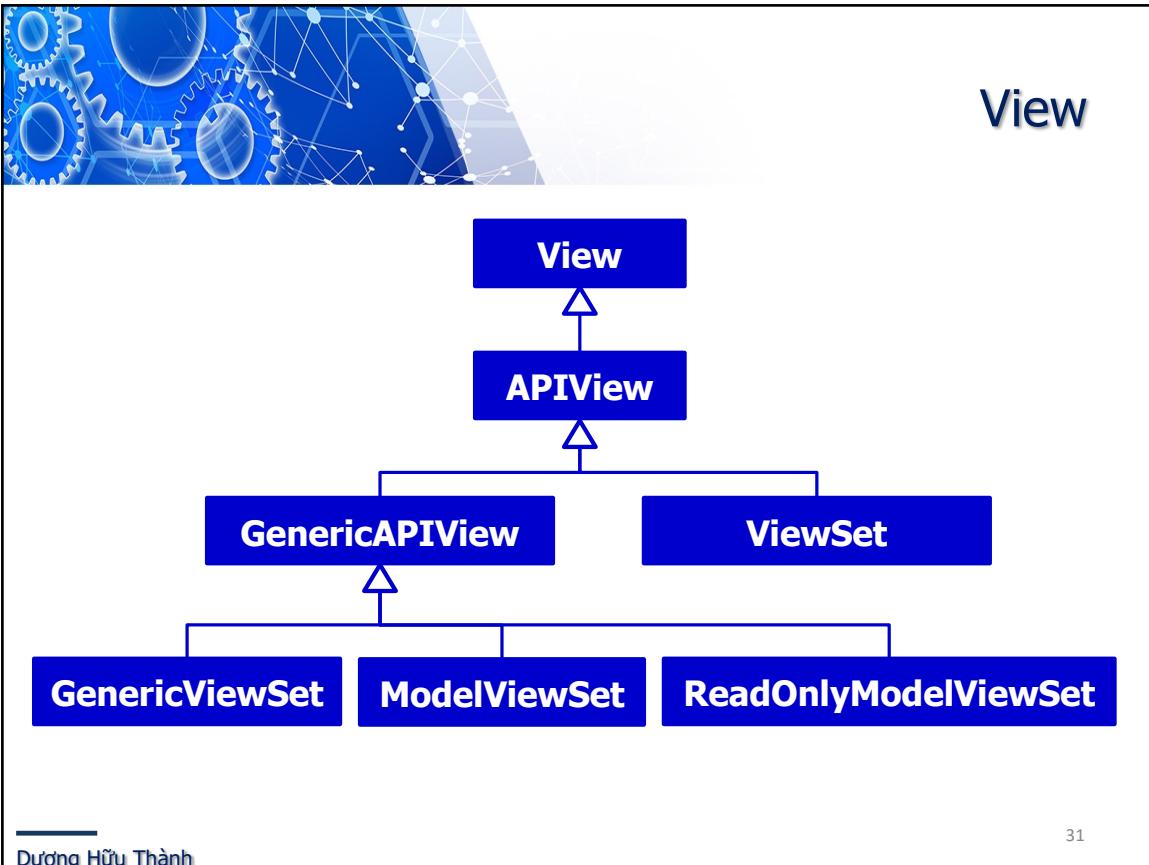


ModelSerializer

```
class UserSerializer(ModelSerializer):  
    class Meta:  
        model = User  
        fields = ['id', 'first_name', 'last_name',  
                  'username', 'password', 'email', 'avatar']  
        extra_kwargs = {'password': {'write_only': True}}  
  
    def create(self, validated_data):  
        data = validated_data.copy()  
  
        user = User(**data)  
        user.set_password(user.password)  
        user.save()  
  
        return user
```

Dương Hữu Thành

30





APIView

```
from rest_framework.views import APIView
from rest_framework.permissions import IsAuthenticated

class CommentAPIView(APIView):
    def get_permissions(self):
        if self.request.method == 'GET':
            return [permissions.AllowAny()]

        return [permissions.IsAuthenticated()]

    def get(self, request, lesson_id):
        pass

    def post(self, request, lesson_id):
        pass
```

Dương Hữu Thành

33



APIView

- Viết phương thức get() của CommentAPIView

```
class CommentAPIView(APIView):
    ...
    def get(self, request, lesson_id):
        comments = Comment.objects.filter(lesson_id=lesson_id)
        serializer = CommentSerializer(comments, many=True)

        return Response(serializer.data,
                        status=status.HTTP_200_OK)
    ...
```

Dương Hữu Thành

34

34

APIView

```
class CommentAPIView(APIView):
    ...
    def post(self, request, lesson_id):
        content = request.data.get('content')
        if content is not None:
            try:
                c = Comment.objects.create(content=content,
                                            user=request.user,
                                            lesson_id=lesson_id)
            except IntegrityError:
                err_msg = "Lesson does not exist!"
            else:
                return Response(CommentSerializer(c).data,
                                status=status.HTTP_201_CREATED)
        else:
            err_msg = "Content is required!!!"

        return Response(data={'error_msg': err_msg},
                        status=status.HTTP_400_BAD_REQUEST)
    ...

```

35

APIView

- Ngoài ra ta có thể viết view bằng function

```
from rest_framework.decorators import api_view

@api_view(['GET', 'POST'])
@permission_classes([IsAuthenticated])
def comment_api_view(request):
    if request.method == 'GET':
        pass
    elif request.method == 'POST':
        pass
```



APIView

- Cập nhật courses/urls.py

```
urlpatterns = [
    ...
    path('lessons/<int:lesson_id>/comments/',
         views.CommentAPIView.as_view()),
]
```

- Khi đó, ta có các API sau:
 - /lessons/{lesson_id}/comments – GET
 - /lessons/{lesson_id}/comments – POST



Generic View

- Generic View giúp phát triển nhanh API view.
- GenericAPIView kế thừa APIView, bổ sung thêm một số hành vi chuẩn như list và detail.



Generic View

- **queryset**: trả về các đối tượng từ view. Queryset sẽ được thực thi một lần khi nó được sử dụng (evaluated) và nó sẽ lưu đệm (cached) cho các lần truy vấn sau. Nếu có ghi đè phương thức `get_queryset()` thì kết quả sẽ ưu tiên cao hơn.
- **serializer_class**: lớp serializer được sử dụng validate, deserialize input và serialize output. Ta có thể ghi đè phương thức `get_serializer_class()` thay thế.

Dương Hữu Thành

39



Generic View

- **lookup_field**: các trường dùng lọc dữ liệu trong các thể hiện model, mặc định là pk.
- **lookup_url_kwarg**: các đối số URL argument.
- **pagination_class**: chỉ định lớp để để phân trang cho kết quả list, mặc định sử dụng giá trị cấu hình của `DEFAULT_PAGINATION_CLASS`. Nếu `paginate_class=None` sẽ không phân trang.
- **get_object()**: trả về thể hiện của đối tượng được sử dụng cho detail.

Dương Hữu Thành

40



Generic View cụ thể

- CreateAPIView
- ListAPIView
- RetrieveAPIView
- DestroyAPIView
- UpdateAPIView
- ListCreateAPIView
- RetrieveUpdateAPIView
- RetrieveDestroyAPIView
- RetrieveUpdateDestroyAPIView

Dương Hữu Thành

41

41



Các lớp Mixins

- Các lớp mixins (`rest_framework.mixins`) cung cấp các action **hiện thực sẵn** các hành vi view cơ bản.
- Một số lớp mixins
 - `ListModelMixin`: hiện thực sẵn `list()`.
 - `CreateModelMixin`: hiện thực sẵn `create()`.
 - `RetrieveModelMixin`: hiện thực sẵn `retrieve()`.
 - `UpdateModelMixin`: hiện thực sẵn `update()` và `update_partial()`
 - `DestroyModelMixin`: hiện thực sẵn `destroy()`.

Dương Hữu Thành

42

42



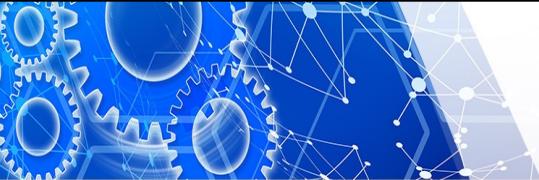
ViewSet

- Django REST framework cho phép **kết hợp logic** của các view liên quan trong một lớp duy nhất gọi là **ViewSet**.
- **ViewSet** **không cung cấp** các phương thức `get()`, `post()`, **thay vào đó** là các phương thức như `list()`, `create()`, `retrieve()`, `destroy()`, `update()`.
- Chú ý: các lớp **ViewSet** **không hiện thực** sẵn các action.

Dương Hữu Thành

43

43



ViewSet

- Các phương thức của **ViewSets**

```
class LessonViewSet(viewsets.ViewSet):  
    def list(self, request):  
        pass  
    def create(self, request):  
        pass  
    def retrieve(self, request, pk=None):  
        pass  
    def update(self, request, pk=None):  
        pass  
    def partial_update(self, request, pk=None):  
        pass  
    def destroy(self, request, pk=None):  
        pass
```

Dương Hữu Thành

44

44

```

class LessonViewSet(viewsets.ViewSet):
    def list(self, request):
        lessons = Lesson.objects.filter(active=True)
        serializer = LessonSerializer(lessons, many=True)

        return Response(data=serializer.data)
    def retrieve(self, request, pk):
        try:
            lesson = Lesson.objects.get(pk=pk)
        except Lesson.DoesNotExist:
            return Http404()

        return Response(LessonSerializer(lesson).data)
    def create(self, request):
        d = request.data
        l = Lesson.objects.create(subject=d['subject'],
                                  content=d['content'],
                                  course_id=d['course_id'])
        serializer = LessonSerializer(l)

        return Response(serializer.data,
                        status=status.HTTP_201_CREATED)

```

45



- Một số thuộc tính trong ViewSet
 - **basename**: tên URL được tạo.
 - **action**: tên các hành vi trong ViewSet như list, retrieve, create.
 - **name**: tên của ViewSet.
 - **description**: mô tả View của ViewSet.
 - **detail**: chỉ định action hiện tại có được cấu hình cho list và detail không.
 - **suffix**: phần đuôi của ViewSet type.



ViewSet

- Kiểm tra quyền trong LessonViewSet như sau:

```
class LessonViewSet(viewsets.ViewSet):  
    def get_permissions(self):  
        if self.action in ['list', 'retrieve']:  
            return [IsAuthenticated()]  
  
        return [IsAdminUser()]  
  
    ...
```

Dương Hữu Thành

47

47



Routers

- Ta **đăng ký ViewSet cho lớp router**, nó sẽ tự động tạo các urlconfs.

```
from django.urls import path, include  
from . import views  
from rest_framework import routers  
  
router = routers.DefaultRouter()  
router.register('courses', views.CourseViewSet)  
router.register('lessons', views.LessonViewSet,  
                basename='lesson')  
  
urlpatterns = [  
    path('', include(router.urls))  
]
```

Dương Hữu Thành

48

48



Routers

- Hai đối số bắt buộc của register()
 - **prefix**: phần tên trước trên URL được sử dụng cho tất cả router, dạng {prefix}/
 - **viewset**: lớp ViewSet.
- Đối số tùy chọn
 - **basename**: tên URL được tạo ra, mặc định dựa trên thuộc tính queryset trong ViewSet.



Routers

- Với khai báo ViewSet này, ta có các API sau:
 - **/lessons/** – GET (list): lấy danh sách bài học, có tên là **lesson-list**.
 - **/lessons/** – POST (create): thêm bài học, có tên là **lesson-add**.
 - **/lessons/{pk}/** – GET (retrieve): xem chi tiết thông tin một bài học, có tên là **lesson-detail**.

The screenshot shows two adjacent browser windows. The left window displays the 'Lesson List' endpoint at `http://127.0.0.1:8000/lessons/`. It shows a JSON response with one item, labeled 'Thực thi list'. The right window displays the 'Lesson Instance' endpoint at `http://127.0.0.1:8000/lessons/1/`. It shows a JSON response with one item, labeled 'Thực thi retrieve'. Both windows have red ovals highlighting specific parts of the JSON responses.

Lesson List

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[{"id": 1, "subject": "Giới thiệu", "content": "Giới thiệu các công nghệ lập trình hiện đại", "course_id": 1, "created_date": "2021-05-30T01:59:15.931582Z", "updated_date": "2021-05-30T07:41:17.071208Z", "tags": [{"id": 1, "name": "nhập môn"}]}
```

Lesson Instance

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[{"id": 1, "subject": "Giới thiệu", "content": "Giới thiệu các công nghệ lập trình hiện đại", "course_id": 1, "created_date": "2021-05-30T01:59:15.931582Z", "updated_date": "2021-05-30T07:41:17.071208Z", "tags": [{"id": 1, "name": "nhập môn"}]}]
```

51

Dương Hữu Thành

51

The screenshot shows a browser window for the 'Lesson List' endpoint at `http://127.0.0.1:8000/lessons/`. A red oval highlights the 'Content' field where a new lesson object is being posted. Below the form, the response shows a successful `HTTP 201 Created` status with the newly created lesson's details, labeled 'Kết quả response'.

Lesson List

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
    "id": 6,
    "subject": "React Native",
    "content": "Phát triển Hybrid App với React Native",
    "course_id": 1,
    "created_date": "2021-06-07T03:24:52.157221Z",
    "updated_date": "2021-06-07T03:24:52.157263Z",
    "tags": []
}
```

Kết quả response

52

Dương Hữu Thành

52



ModelViewSet

- ModelViewSet cung cấp **một số xử lý mặc định** cho ViewSet.
- Ví dụ tạo ModelViewSet cho Lesson.

```
class LessonViewSet(viewsets.ModelViewSet):  
    serializer_class = LessonSerializer  
    queryset = Lesson.objects.filter(active=True)
```

- **Bổ sung các urls** của ViewSet này vào route của REST framework.



ModelViewSet

- Các API sau được tạo ra tự động
 - /lessons/ - GET
 - /lessons/ - POST
 - /lessons/{id}/ - GET
 - /lessons/{id}/ - PUT
 - /lessons/{id}/ - PATCH
 - /lessons/{id}/ - DELETE

ModelViewSet

- Thêm action vào ViewSet, bổ sung thêm API ẩn bài học (cập nhật active thành False)

```
from rest_framework.decorators import action

class LessonViewSet(viewsets.ModelViewSet):
    @action(methods=['post'], detail=True)
    def hide_lesson(self, request, pk=None):
        ...
```

- Mặc định API được tạo (ta có thể đổi thông tin URL bằng thuộc tính **url_path** và **url_name**)
 - /lessons/{pk}/hide_lesson/ - POST



ModelViewSet

```
class LessonViewSet(viewsets.ModelViewSet):
    ...
    @action(methods=['post'], detail=True,
            name='Hide this lesson',
            url_path='hide-lesson',
            url_name='hide-lesson')
    def hide_lesson(self, request, pk=None):
        try:
            lesson = Lesson.objects.get(pk=pk)
            lesson.active = False
            lesson.save()
        except Course.DoesNotExist:
            return Response(status=status.HTTP_400_BAD_REQUEST)

        serializer = LessonSerializer(lesson)
        return Response(serializer.data,
                        status=status.HTTP_200_OK)
```

ModelViewSet

- Truy cập vào một bài học cụ thể

The screenshot shows a browser window titled "Lesson Instance – Django REST". The URL in the address bar is "127.0.0.1:8000/lessons/1/". The page title is "Django REST framework" and the sub-page title is "Lesson Instance". There are several buttons at the top right: "Extra Actions", "DELETE", "OPTIONS", and "GET". A red circle highlights the "GET /lessons/1/" button. Another red circle highlights the "Hide this lesson" button under the "Extra Actions" dropdown. Below the buttons, there is some API documentation and a JSON response. The JSON response includes fields like "id", "subject", "content", "course_id", "created_date", "updated_date", and "tags". The "tags" field contains a single entry for "nhập môn". At the bottom left, it says "Dương Hữu Thành". On the right side, there is a small number "57".

57

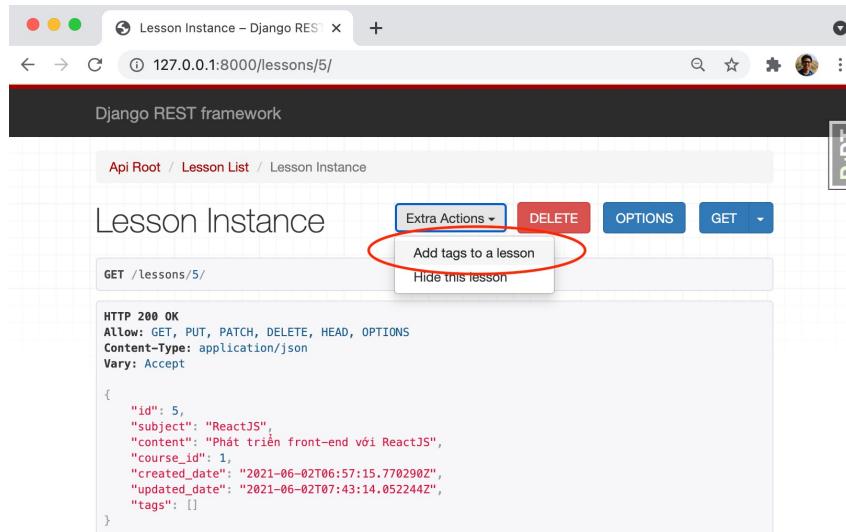
ModelViewSet

The screenshot shows a browser window titled "Hide this lesson – Django REST". The URL in the address bar is "127.0.0.1:8000/lessons/1/hide-lesson/". The page title is "Django REST framework" and the sub-page title is "Hide this lesson". There are two buttons at the top right: "Extra Actions" and "OPTIONS". A red circle highlights the "GET /lessons/1/hide-lesson/" button. Below the buttons, there is some API documentation and a JSON response. The JSON response includes a "detail" field with the message "Method \"GET\" not allowed.". At the bottom, there is a form with "Media type: application/json" and a "Content:" field containing a JSON object with empty fields for "subject", "content", and "tags". A red box highlights the "POST" button. Below the form, there is a note in Vietnamese: "API không yêu cầu thông tin gì trong body, nên chỉ cần click POST để thực thi bài viết". At the bottom left, it says "Dương Hữu Thành". On the right side, there is a small number "58".

58

ModelViewSet

- Thêm action cho phép gán tag vào cho lesson.



59

Dương Hữu Thành

59

```
class LessonViewSet(viewsets.ModelViewSet):  
    @action(methods=['post'], detail=True,  
            name="Add tags to a lesson",  
            url_path='add-tags-to-lesson',  
            url_name='add-tags')  
    def add_tags_to_lesson(self, request, pk):  
        try:  
            lesson = Lesson.objects.get(pk=pk)  
  
            tags = request.data.get('tags')  
            for tag in tags.split(','):  
                t, _ = Tag.objects.get_or_create(name=tag.strip())  
                lesson.tags.add(t)  
            lesson.save()  
        except Lesson.DoesNotExist | KeyError:  
            return Response(status=status.HTTP_404_NOT_FOUND)  
  
        serializer = LessonSerializer(lesson)  
        return Response(serializer.data,  
                        status=status.HTTP_200_OK)
```

Dương Hữu Thành

60

The screenshot shows a browser window with a blue gear and network background. The title bar says "Add tags to a lesson - Django". The address bar shows "127.0.0.1:8000/lessons/5/add-tags-to-lesson/". The main content area is titled "Django REST framework" and shows the URL "GET /lessons/5/add-tags-to-lesson/". Below it, an error message for "HTTP 405 Method Not Allowed" is displayed, indicating that only POST and OPTIONS methods are allowed. A "Content" field contains the JSON payload `{"tags": "reactjs, redux"}`. A red oval highlights this content field. At the bottom right of the form is a "POST" button.

61

61

The screenshot shows a browser window with a blue gear and network background. The title bar says "Add tags to a lesson - Django". The address bar shows "127.0.0.1:8000/lessons/5/add-tags-to-lesson/". The main content area is titled "Django REST framework" and shows the URL "POST /lessons/5/add-tags-to-lesson/". Below it, a success message "HTTP 200 OK" is displayed, along with the allowed methods: POST, OPTIONS. A "Content-Type: application/json" header is also shown. A large red oval highlights the "Response Data" section, which displays the JSON response: { "id": 5, "subject": "ReactJS", "content": "Phát triển front-end với ReactJS", "course_id": 1, "created_date": "2021-06-02T06:57:15.770290Z", "updated_date": "2021-06-07T06:58:42.556182Z", "tags": [{ "id": 5, "name": "reactjs" }, { "id": 6, "name": "redux" }] }

62

62



Parsers

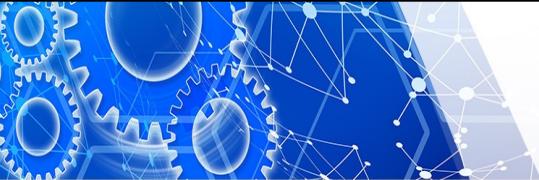
- REST cung cấp nhiều lớp Parser cho phép xử lý các request với nhiều media type khác nhau.
- Khi nhận request.data, REST dựa trên Content-Type của request để xác định sử dụng parser nào để parse nội dung request.
- Ta có thể thiết lập parser mặc định toàn cục cho ứng dụng trong settings giống như sau:

```
REST_FRAMEWORK = {  
    'DEFAULT_PARSER_CLASSES': [  
        'rest_framework.parsers.JSONParser',  
    ]  
}
```

Dương Hữu Thành

63

63



Parsers

- Ví dụ sử dụng FileUploadParser cho view đăng ký người dùng vì cần upload ảnh đại diện.

```
from rest_framework.parsers import MultiPartParser  
  
class UserViewSet(viewsets.ViewSet,  
                  generics.CreateAPIView):  
    serializer_class = UserSerializer  
    queryset = User.objects.filter(is_active=True)  
    parser_classes = [MultiPartParser, ]
```

Dương Hữu Thành

64

64



Authentication

- Authentication là cơ chế kết hợp các request nhận được với tập các thông tin chứng thực.
- Các chính sách permission và throttling sử dụng các thông tin chứng thực để quyết định request có quyền truy cập không.
- Authentication luôn **chạy khi bắt đầu view** và **trước khi kiểm tra permission** và throttling.

Dương Hữu Thành

65

65



Authentication

- **request.user** là thể hiện của lớp User thuộc gói contrib.auth.
- **request.auth** chứa thông tin chứng thực khác, chẳng hạn lưu token.

Dương Hữu Thành

66

66

Thực hiện Authentication

- Các loại chứng thực (authentication schemes) được định nghĩa danh sách lớp python.
- REST framework sẽ **chứng thực mỗi lớp trong danh sách và thiết lập giá trị request.user và request.auth** cho lớp đầu tiên được chứng thực thành công.
- Nếu không có lớp nào chứng thực thành công
 - request.user là thể hiện của `django.contrib.auth.models.AnonymousUser`
 - `Request.auth = None`

Dương Hữu Thành

67

67

Thực hiện Authentication

- Cấu hình **DEFAULT_AUTHENTICATION_CLASSES** trong biến REST_FRAMEWORK trong settings.py để **chỉ định các lớp mặc định** dùng chứng thực.

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.BasicAuthentication',
        'rest_framework.authentication.TokenAuthentication'
    ]
}
```

Dương Hữu Thành

68

68

Thực hiện Authentication

- Ngoài ra, ta có thể thiết lập các lớp chứng thực trong các lớp view thông qua thuộc tính **authentication_classes**

```
class UserView(APIView):  
    authentication_classes = [BasicAuthentication,  
                             TokenAuthentication]  
    permission_classes = [IsAdminUser]  
  
    def get(self, request):  
        pass
```

Thực hiện Authentication

- Nếu một request không được chứng thực thì có 2 error code sau đây phù hợp
 - **HTTP 401 – Unauthorized**: luôn có header WWW-Authenticate hướng dẫn client làm thế nào chứng thực.
 - **HTTP 403 – Permission Denied**
- Chú ý khi request được chứng thực thành công, nhưng **không có quyền** truy cập tài nguyên nào đó thì **HTTP 403** luôn được sử dụng.



Các lớp Authentication

- **BasicAuthentication**: chứng thực người dùng thông qua **username và password**. Nếu chứng thực thành công thì
 - `request.user` là thể hiện của User của Django.
 - `request.auth = None`
 - Nếu request không được chứng thực thì **HTTP 401 Unauthorized** sẽ được trả về.



Các lớp Authentication

- **TokenAuthentication**: chứng thực người dùng thông qua **token**, phù hợp cho ứng dụng theo **kiến trúc client-server**.
- Để sử dụng lớp chứng thực này, ta cần cấu hình thêm cho biến `INSTALLED_APP` như sau:

```
INSTALLED_APPS = [  
    ...  
    'rest_framework.authtoken'  
]
```

- Thực thi lệnh `migrate`.

Các lớp Authentication

- Tạo token cho user

```
from rest_framework.authtoken.models import Token  
  
token = Token.objects.create(user=...)  
print(token.key)
```

- Phía client chứng thực thiết lập Authorization trong **headers** như sau:

```
Authorization: Token <token-value>
```

- Khi đó request.auth là thể hiện của `rest_framework.authtoken.models.Token`

Các lớp Authentication

- **SessionAuthentication**: lớp chứng thực này sử dụng session của django. Lớp chứng thực này phù hợp cho client dùng ajax chạy trong cùng ngữ cảnh session của website.
- **RemoteUserAuthentication**: lớp chứng thực này cho phép ủy quyền chứng thực cho web server, thiết lập cho biến môi trường `REMOTE_USER`.



OAuth2

- OAuth là một giao thức mở và an toàn để **chứng thực** người dùng **giữa các dịch vụ** liên quan.
- OAuth **ủy quyền** một dịch vụ được phép truy cập những tài nguyên từ một dịch vụ khác đại diện cho người dùng, mà **không cần** cung cấp thông tin như username, password.
- OAuth2 là phiên bản mới nhất của giao thức OAuth.



OAuth2

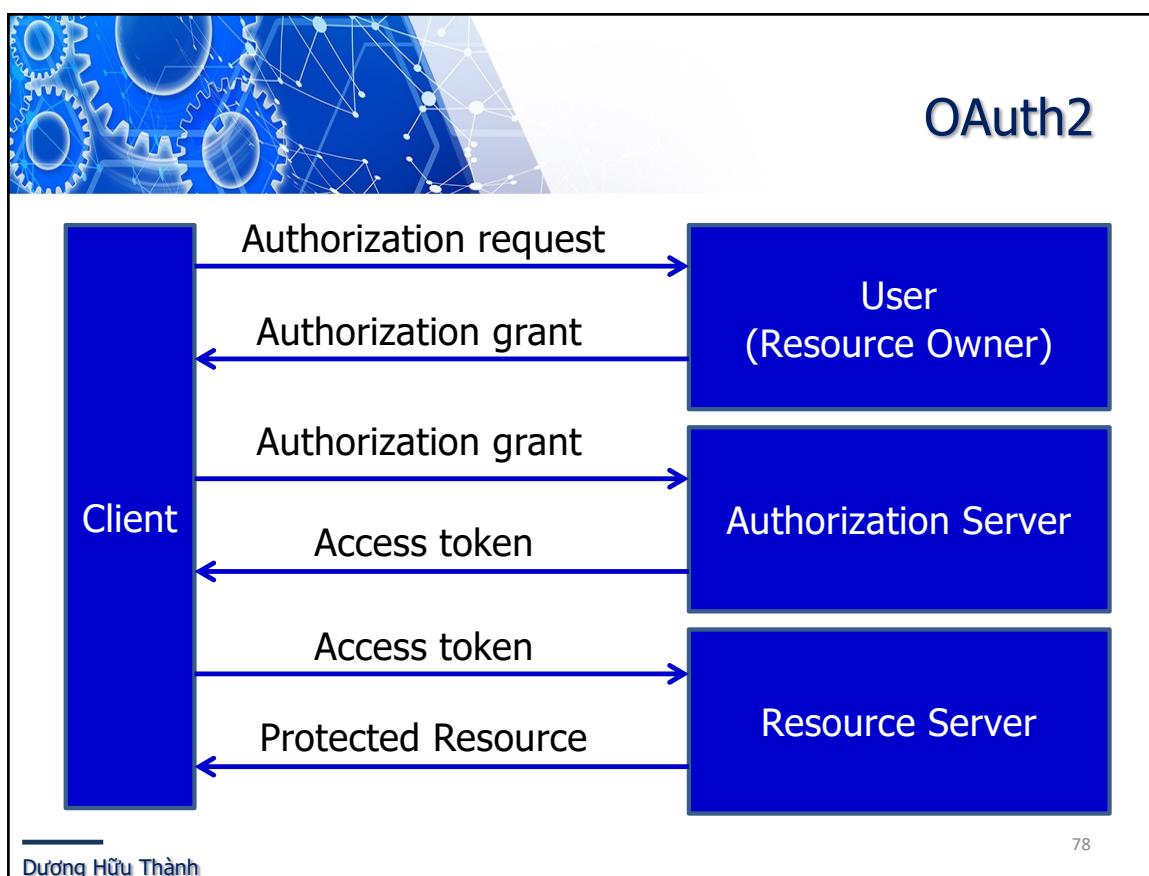
- **Client:** ứng dụng **thực hiện hành vi** đại diện người dùng truy cập các tài nguyên.
- **Resource Owner/User:** là **người dùng ủy quyền** cho ứng dụng **truy cập** vào các tài nguyên được bảo vệ bởi nhà cung cấp khác.
- **Authorization Server:** server chứng thực yêu cầu từ phía client.
- **Resource Server:** server chứa các tài nguyên được bảo vệ.

OAuth2

- Các bước thực hiện giao tiếp giữa client và server theo giao thức OAuth2:
 - Authorize: chứng thực yêu cầu quyền truy cập từ client.
 - Yêu cầu access token.
 - Truy cập các tài nguyên được bảo vệ.

Dương Hữu Thành

77



Dương Hữu Thành

78

78



OAuth2

- Trước khi sử dụng OAuth2 để chứng thực, người phát triển (developer) phải **đăng ký một application** với OAuth2 provider.
- Sau khi đăng ký xong sẽ được cung cấp thông tin: **client ID, client secret, authorization server URL, access token URL, resource server URL.**

Dương Hữu Thành

79

79



OAuth2

- Client ID: thông tin công khai định danh ứng dụng, giống như username.
- Client secret: thông tin riêng tư gửi kèm lên server, giống như password.
- Authorization server URL: URL để client thực hiện các yêu cầu (request).
- Access token URL: URL để client yêu cầu access token.
- Resource server URL: URL để client truy cập các tài nguyên được bảo vệ.

Dương Hữu Thành

80

80



Django OAuth Toolkits

- Django OAuth Toolkit cung cấp tất cả các endpoint, dữ liệu và logic cần thiết để thực hiện giao thức OAuth2 cho django project.
- Cài đặt

```
pip install django-oauth-toolkit
```



Django OAuth Toolkits

- Cập nhật biến INSTALLED_APP trong settings.py

```
INSTALLED_APPS = (
    ...
    'oauth2_provider',
)
```

- Bổ sung thông tin cấu hình cho biến REST_FRAMEWORK trong settings.py

```
REST_FRAMEWORK = {
    ...
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'oauth2_provider.contrib.rest_framework.OAuth2Authentication',
    )
}
```



Django OAuth Toolkits

- Cập nhật urls cho URLConfig của project

```
from django.urls import include, path  
  
urlpatterns = [  
    ...  
    path('o/', include('oauth2_provider.urls',  
                       namespace='oauth2_provider')),  
]
```

- Thực thi migrate

```
python manage.py migrate oauth2_provider
```

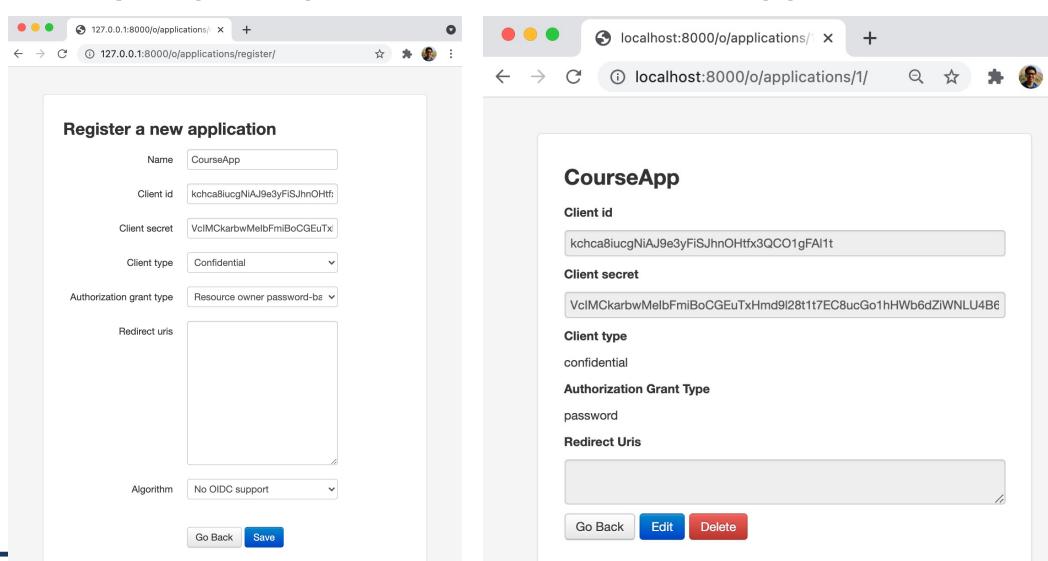
Dương Hữu Thành

83



Django OAuth Toolkits

- Truy cập: <http://127.0.0.1:8000/o/applications/>



The image shows two screenshots of a web browser. The left screenshot shows the 'Register a new application' form with fields: Name (CourseApp), Client id (kchca8iucgNiAJ9e3yFiSJhnOHttp), Client secret (VclIMCkarbwMelbFmiBoCGEuTx), Client type (Confidential), Authorization grant type (Resource owner password-based), Redirect uris (empty), Algorithm (No OIDC support), and buttons (Go Back, Save). The right screenshot shows the details of the registered application 'CourseApp' with fields: Client id (kchca8iucgNiAJ9e3yFiSJhnOHttp3QCO1gFAI1t), Client secret (VclIMCkarbwMelbFmiBoCGEuTxHmd9l28t117EC8ucGo1hHWb6dZWNLU4B6), Client type (confidential), Authorization Grant Type (password), and Redirect Uris (empty). Buttons (Go Back, Edit, Delete) are at the bottom.

84



Django OAuth Toolkits

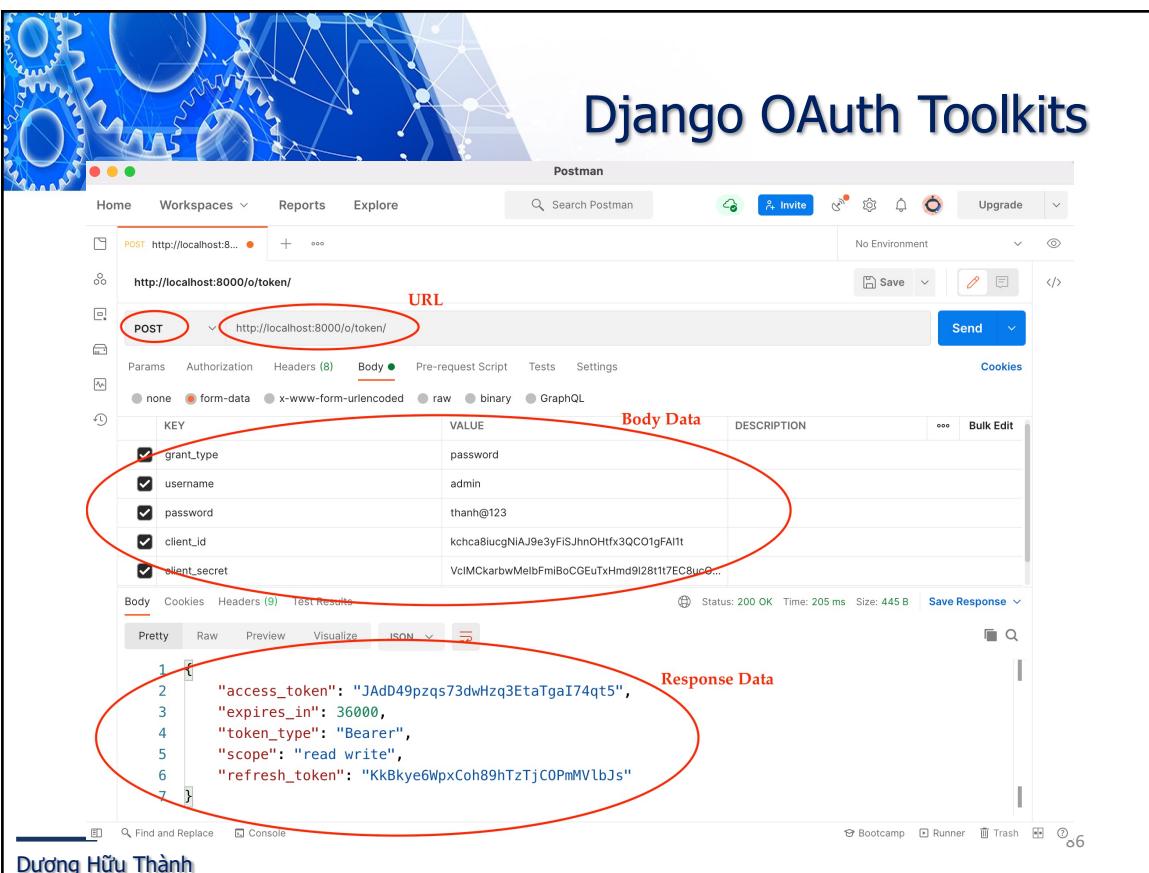
- Yêu cầu lấy access token
 - Url: **/o/token/**
 - Method: POST
 - Body data

```
{
  "grant_type": "password",
  "username": "<username>",
  "password": "<password>",
  "client_id": "<client-id>",
  "client_secret": "<client-secret>"}
```

Dương Hữu Thành

85

85



POST http://localhost:8000/o/token/

Body Data

KEY	VALUE	DESCRIPTION	Bulk Edit
grant_type	password		
username	admin		
password	thanh@123		
client_id	kchca8lucgNIAJ9e3yFiSJhnOHTfx3QCO1gFA1t		
client_secret	VclIMCkarbwMeibFmIBoCGEuTxHmd9l28t1t7EC8u... C...		

Response Data

```

1  "access_token": "JAdD49pzqs73dwHzq3EtaTgaI74qt5",
2  "expires_in": 36000,
3  "token_type": "Bearer",
4  "scope": "read write",
5  "refresh_token": "KkBkye6WpxCoh89hTzTjCOPmMVlbJs"
6
7
```

Dương Hữu Thành

86



Django OAuth Toolkits

- Sau khi có **access token**, ta có thể sử dụng để yêu cầu truy cập một số **tài nguyên yêu cầu chứng thực** mà không cần gửi username, password.
- Trong HTTP header của mỗi request phải gửi kèm Authorization, giá trị của nó có thể là **Token** hoặc **Bearer** phụ thuộc OAuth2 provider.
- Cú pháp
 - Authorization: Bearer <access-token>

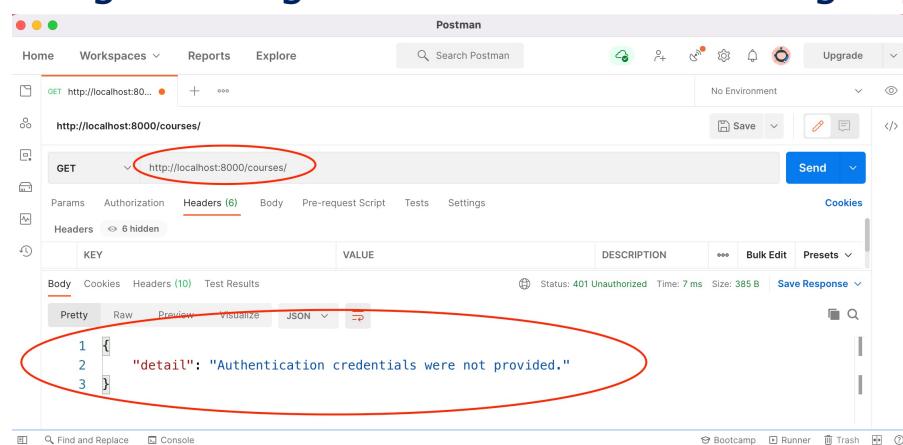
Dương Hữu Thành

87



Django OAuth Toolkits

- Phần trước ta đã cấu hình cho API /courses/ chỉ được truy cập khi đăng nhập, giờ truy cập vào không có thông tin access token để chứng thực.



Dương Hữu Thành

88

88

Thông tin access token đặt trong
headers của request

```
1 {
2     "count": 8,
3     "next": "http://localhost:8000/courses/?page=2",
4     "previous": null,
5     "results": [
6         {
7             "id": 8,
8             "subject": "Cơ sở lập trình",
9             "created_date": "2021-06-01T03:14:17.053Z",
10            "category_id": 1
11        },
12    ]
}
```

Dương Hữu Thành

89

CORS

- CORS (**Cross-origin resource sharing**) là cơ chế cho phép **giới hạn các tài nguyên** trên trang web **được yêu cầu** truy cập từ **domain khác**.
- CORS cho phép trình duyệt và server xác định **được request cross-origin** có an toàn không.

Dương Hữu Thành

90

90



CORS

- Cài cors middleware

```
pip install django-cors-headers
```
- Bổ sung biến INSTALLED_APP trong settings.py

```
INSTALLED_APP = [  
    ...  
    'corsheaders'  
]
```



CORS

- Cập nhật biến MIDDLEWARE của settings.py giá trị corsheaders.middleware.CorsMiddleware. Giá trị này nên đặt trước các middleware có thể tạo ra response.

```
MIDDLEWARE = [  
    'corsheaders.middleware.CorsMiddleware',  
    ...  
]
```

- Thiết lập giá trị biến sau trong settings.py

```
CORS_ALLOW_ALL_ORIGINS = True
```



JSON Web Token

- JSON Web Token (JWT) là tiêu chuẩn chứng thực mới dựa trên token, nhưng không giống với TokenAuthentication có sẵn, JWT **không sử dụng CSDL** để kiểm tra token.
- Ở đây minh họa sử dụng JWT thông qua thư viện Simple JWT.

93

Dương Hữu Thành

93



Simple JWT

- Cài đặt

```
pip install djangorestframework-simplejwt
```

- Chỉ định lớp chứng thực trong settings.py

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        ...
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    )
}
```

94

Dương Hữu Thành

94

Simple JWT

- Cập nhật urls cho URLConfig của project

```
from rest_framework_simplejwt.views import (
    TokenObtainPairView,
    TokenRefreshView
)

urlpatterns = [
    ...
    path('api/token/', TokenObtainPairView.as_view(),
         name='token_obtain_pair'),
    path('api/token/refresh/', TokenRefreshView.as_view(),
         name='token_refresh'),
]
```

Dương Hữu Thành

95

95

Simple JWT

The screenshot shows the Postman interface with the following details:

- Method:** POST (circled in red)
- URL:** http://127.0.0.1:8000/api/token/ (circled in red)
- Body Data:** Form-data (circled in red)

KEY	VALUE	DESCRIPTION
username	admin	
password	thanh@123	
- Response Data:** JSON output showing two tokens:

```
1 "refresh": "eyJ0eXAi0iJKV1QiLCJhbGciOiJIUzI1NiJ9.
eyJ0b2tlbl90eXBLIjoicmVmcmVzaC1sImV4cCI6MTYyMjg1ODg2NCwianRpIjoiMDVkdZDA0NWE5ZWm4NGQ5ZWFiIN
jQwZjU4NTYyNWIyZDQjLCJ1c2VvXlkIjoxfQ.iRKeV7RpMnh2aB1XzUSq1b6jDysjh1b-bGLhJ1DpLo",
"access": "eyJ0eXAi0iJKV1QiLCJhbGciOiJIUzI1NiJ9.
eyJ0b2tlbl90eXBLIjoivWnjZNzIwiwZkhwIjoxNjIyNzcyNzY0LCJqdGkiOiI1NzkzYTZiNWQ1Yzg0YWVjYmQyO
WI3NmVhZGUyZjI2YiIsInVzZXJfaWQ0jF9.IpeR4GLrn0fXz7C6Nv4gkJz27bbjxKdn-FXuz4mJks8"
```

Dương Hữu Thành

96

96

Simple JWT

Postman

GET http://127.0.0.1:8000/courses/

Headers (7)

KEY	VALUE
Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoiYWNj...

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

Status: 200 OK Time: 109 ms Size: 965 B Save Response

Dương Hữu Thành

97

Tích hợp Swagger

- Cài đặt drf-yasg (Yet Another Swagger Generator)

```
pip install drf-yasg
```
- Cập nhật biến INSTALLED_APP

```
INSTALLED_APPS = [  
    ...  
    'django.contrib.staticfiles',  
    'drf_yasg',  
]
```

Dương Hữu Thành

98



Tích hợp Swagger

- Cập nhật urls cho URLConfs của project

```
from rest_framework import permissions
from drf_yasg.views import get_schema_view
from drf_yasg import openapi

schema_view = get_schema_view(
    openapi.Info(
        title="Course API",
        default_version='v1',
        description="APIs for CourseApp",
        contact=openapi.Contact(email="thanh.dh@ou.edu.vn"),
        license=openapi.License(name="Dương Hữu Thành@2021"),
    ),
    public=True,
    permission_classes=(permissions.AllowAny,),
)
```

99



Tích hợp Swagger

- Cập nhật urls cho URLConf của project

```
...
urlpatterns = [
    ...
    re_path(r'^swagger(?P<format>\.json|\.yaml)$',
            schema_view.without_ui(cache_timeout=0),
            name='schema-json'),
    re_path(r'^swagger/$',
            schema_view.with_ui('swagger', cache_timeout=0),
            name='schema-swagger-ui'),
    re_path(r'^redoc/$',
            schema_view.with_ui('redoc', cache_timeout=0),
            name='schema-redoc')
]
```

The screenshot shows the Swagger UI interface for a Course API version 1. The title bar says "Tích hợp Swagger". The URL in the address bar is "http://127.0.0.1:8000/swagger/#". The main content area displays the "Course API v1" documentation. It includes sections for "api" and "courses". Under "api", there are two POST methods: "/api/token/" (with a lock icon) and "/api/token/refresh/" (with a lock icon). Under "courses", there are two methods: "GET /courses/" (with a lock icon) and "POST /courses/" (with a lock icon). A sidebar on the left shows the user "Dương Hữu Thành". At the bottom right, the number "101" is displayed.

101

The screenshot shows the ReDoc API documentation for the "courses" endpoint. The title bar says "Tích hợp Swagger". The URL in the address bar is "http://127.0.0.1:8000/redoc/#tag/courses". The left sidebar lists various endpoints under "api" and "courses". The "courses" section is expanded, showing methods: "GET courses_list", "POST courses_create", "GET courses_read", "PUT courses_update", "PATCH courses_partial_update", and "DELETE courses_delete". The "courses_list" endpoint is selected. On the right, it shows "AUTHORIZATIONS: Basic" and "QUERY PARAMETERS: page" (described as "A page number within the paginated result set"). Below this, the "Responses" section shows the "200" response with a "RESPONSE SCHEMA: application/json". The schema is defined as an object with properties: "count" (type: integer, required), "next" (type: string <uri> Nullable), "previous" (type: string <uri> Nullable), and "results" (type: array of objects (Course), required).

102

Tích hợp Swagger

- Ta có thể thiết lập ẩn một api để không đưa vào swagger bằng cách thiết lập thuộc tính `swagger_schema = None`

```
class UserList(APIView):  
    swagger_schema = None  
    ...
```

Dương Hữu Thành

103

103

Tích hợp Swagger

```
from drf_yasg.utils import swagger_auto_schema  
from drf_yasg import openapi  
  
class LessonViewSet(viewsets.ModelViewSet):  
    @swagger_auto_schema(  
        operation_description='Add tags to a lesson',  
        responses={  
            status.HTTP_200_OK: LessonSerializer()  
        }  
    )  
    @action(methods=['post'], detail=True,  
            name="Add tags to a lesson",  
            url_path='add-tags-to-lesson',  
            url_name='add-tags')  
    def add_tags_to_lesson(self, request, pk):  
        ...
```

Dương Hữu Thành

104

104

Tích hợp Debug Toolbar

- Cài đặt

```
pip install django-debug-toolbar
```

- Cập nhật biến INSTALLED_APP và STATIC_URL

```
INSTALLED_APPS = [  
    # ...  
    'django.contrib.staticfiles',  
    # ...  
    'debug_toolbar',  
]  
  
STATIC_URL = '/static/'
```

Dương Hữu Thành

105

105

Tích hợp Debug Toolbar

- Debug Toolbar chủ yếu được thực hiện trong middleware. Trong settings.py bật middleware

```
MIDDLEWARE = [  
    ...  
    'debug_toolbar.middleware.DebugToolbarMiddleware',  
]
```

- Debug Toolbar chỉ được hiển thị trên các địa chỉ IP có trong danh sách INTERNAL_IPS của tập tin cấu hình.

```
INTERNAL_IPS = [  
    '127.0.0.1'  
]
```

Dương Hữu Thành

106

106

Tích hợp Debug Toolbar

- Cập nhật urls cho URLConfig của project

```
import debug_toolbar
from django.conf import settings
from django.urls import include, path

urlpatterns = [
    ...
    path('__debug__/', include(debug_toolbar.urls))
]
```

Dương Hữu Thành

107

107

Tích hợp Debug Toolbar

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Courses > Courses

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

COURSES

Categories [+ Add](#)

Courses [+ Add](#)

Lessons [+ Add](#)

DJANGO OAUTH TOOLKIT

Access tokens [+ Add](#)

Applications [+ Add](#)

Grants [+ Add](#)

Id tokens [+ Add](#)

Refresh tokens [+ Add](#)

Select course to change

Action: Go 0 of 8 selected

COURSE

Cơ sở lập trình

Kỹ thuật lập trình

Cấu trúc dữ liệu và giải thuật

Công nghệ phần mềm

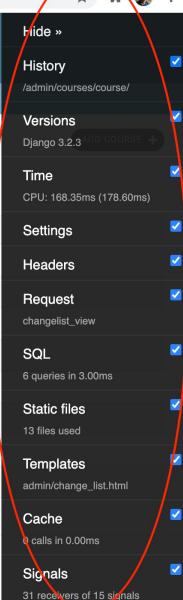
Kiểm thử phần mềm

Thiết kế Web

Lập trình Java

Các công nghệ lập trình hiện đại

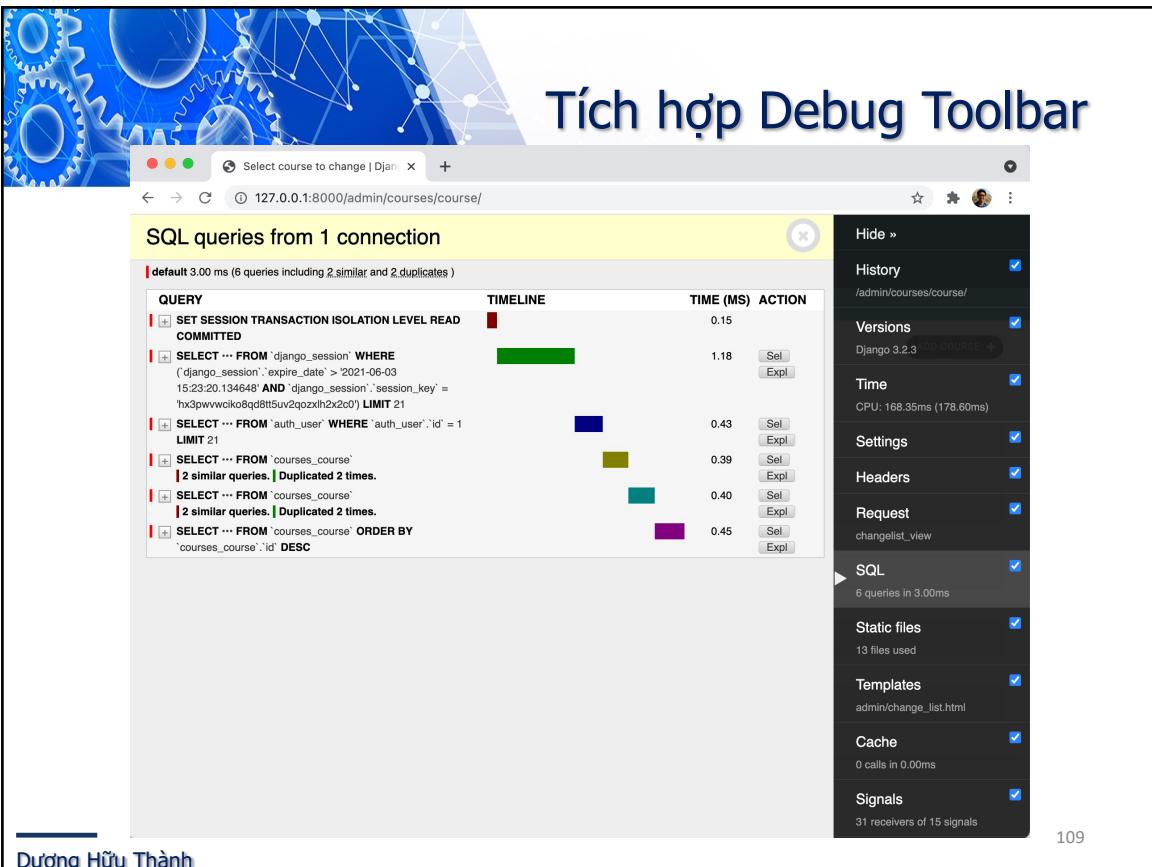
8 courses



Dương Hữu Thành

108

108



109



110