# 🔥 `DestroyAPIView` dans Django REST Framework (DRF)

`DestroyAPIView` est une **vue générique** fournie par DRF qui permet de **supprimer un objet** spécifique dans une API REST. Elle gère automatiquement la méthode HTTP **DELETE**.

---

## 📌 Comment ça marche ?

### 1️⃣ Définition de `DestroyAPIView`

`DestroyAPIView` hérite de `GenericAPIView` et ajoute la méthode `destroy()`, qui supprime un objet et renvoie une réponse `204 No Content`.

### 2️⃣ Exemple d'utilisation

**✅ Modèle Django :**

```
from django.db import models

class Service(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField()
```

**✅ Serializer :**

```
from rest_framework import serializers
from .models import Service

class ServiceSerializer(serializers.ModelSerializer):
    class Meta:
        model = Service
        fields = '__all__'
```

**✅ Vue avec `DestroyAPIView` :**

```
from rest_framework import generics
from .models import Service

class ServiceDeleteView(generics.DestroyAPIView):
    queryset = Service.objects.all()
    lookup_field = 'id'  # Par défaut, c'est `pk`
```

✅ **URLS :**

```
from django.urls import path
from .views import ServiceDeleteView

urlpatterns = [
    path('services/<int:id>/delete/', ServiceDeleteView.as_view(),
name='service-delete'),
]
```

---

## 🛠️ Comment fonctionne `DestroyAPIView` ?

1️⃣ L'utilisateur envoie une requête **DELETE** vers `/services/1/delete/`.
2️⃣ La vue `ServiceDeleteView` cherche l'objet `Service` avec `id=1`.
3️⃣ Si trouvé, l'objet est supprimé et **DRF renvoie une réponse `204 No Content`**.
4️⃣ Si l'objet n'existe pas, DRF retourne une erreur **404 Not Found**.

---

## ⚠️ Personnalisation avec des permissions

Tu peux restreindre la suppression à certains utilisateurs avec `permission_classes` :

```
from rest_framework.permissions import IsAdminUser

class ServiceDeleteView(generics.DestroyAPIView):
    queryset = Service.objects.all()
    lookup_field = 'id'
    permission_classes = [IsAdminUser]  # Seuls les admins peuvent
supprimer
```

---

## 🏆 Quand utiliser `DestroyAPIView` ?

✔️ Lorsque tu veux **uniquement** gérer la suppression d'un objet.
✔️ Si tu veux une vue REST **simple et rapide** pour DELETE.
✔️ Si tu veux profiter des **permissions DRF** sans tout coder à la main.

👉 **Dans ton projet de réservation de services, tu peux l'utiliser pour permettre aux administrateurs de supprimer des réservations ou des services.**

### 🔍 `lookup_field` dans Django REST Framework (DRF)

Dans **Django REST Framework**, `lookup_field` est un attribut utilisé dans les vues basées sur `GenericAPIView` (et ses sous-classes comme `RetrieveAPIView`, `UpdateAPIView`, `DestroyAPIView`, etc.). Il permet de **spécifier quel champ** du modèle doit être utilisé pour rechercher un objet.

---

## 📌 Par défaut : `pk` (Primary Key)

Quand tu utilises une vue générique comme `RetrieveAPIView` ou `DestroyAPIView`, DRF utilise **par défaut** `pk` (la clé primaire) pour chercher l'objet.

- **Exemple sans `lookup_field` (utilisation de `pk`) :**

```
class ServiceDeleteView(generics.DestroyAPIView):
    queryset = Service.objects.all()
    serializer_class = ServiceSerializer
```

➡️ Dans ce cas, l'URL ressemblera à :

```
DELETE /services/1/  # Suppression du service avec id=1
```

---

## 📌 Changer le champ de recherche avec `lookup_field`

Tu peux remplacer `pk` par un **autre champ** de ton modèle, comme `slug` ou `nom`.

- **Exemple avec `lookup_field="slug"` :**

```
class ServiceDeleteView(generics.DestroyAPIView):
    queryset = Service.objects.all()
    serializer_class = ServiceSerializer
    lookup_field = 'slug'  # Recherche par slug au lieu de pk
```

➡️ L'URL devient :

```
DELETE /services/nettoyage-maison/  # Suppression par slug
```

👉 Ici, DRF cherchera `Service.objects.get(slug="nettoyage-maison")` au lieu de `pk=1`.

---

## 🎯 Cas d'utilisation

✅ **Recherche par un autre identifiant unique** (ex : `username`, `email`, `slug`, etc.)
✅ **Meilleure lisibilité des URLs** (`/services/nettoyage-maison/` au lieu de `/services/1/`)
✅ **Cas où `pk` n'est pas le meilleur choix** (ex : un modèle sans clé primaire explicite)

🛠️ **Tu veux l'utiliser dans ton projet de réservation ?**
Tu peux rechercher un service par **son nom unique** au lieu de son ID, par exemple :

```python
class BookingRetrieveView(generics.RetrieveAPIView):
    queryset = Booking.objects.all()
    serializer_class = BookingSerializer
    lookup_field = 'reference'  # Recherche une réservation par référence
unique
```

➡️ **URL:** `/bookings/ABC123/` au lieu de `/bookings/1/`

---