

# ◆ Rôle de la méthode `perform_create` dans une vue générique Django REST Framework (DRF)

---

Dans DRF, la méthode `perform_create` est utilisée dans les vues génériques basées sur `CreateAPIView` pour personnaliser la création d'un objet avant de l'enregistrer dans la base de données.

---

## ◆ Pourquoi `perform_create` ?

Lorsqu'on utilise une vue générique comme `CreateAPIView`, DRF gère la création de l'objet automatiquement en appelant la méthode `serializer.save()`.

Cependant, `perform_create` permet d'ajouter une logique personnalisée avant d'enregistrer l'objet. Par exemple, on peut :

- Assigner un utilisateur connecté (`request.user`)
  - Modifier certaines valeurs avant la sauvegarde
  - Déclencher des actions supplémentaires (ex : envoyer un email, logger une action, etc.)
- 

## ◆ Exemple 1 : Assigner l'utilisateur connecté

Si on a un modèle `Article` et qu'on veut associer l'auteur à l'utilisateur connecté :

```
from rest_framework.generics import CreateAPIView
from myapp.models import Article
from myapp.serializers import ArticleSerializer

class ArticleCreateView(CreateAPIView):
    queryset = Article.objects.all()
    serializer_class = ArticleSerializer

    def perform_create(self, serializer):
        # Associer l'auteur de l'article à l'utilisateur connecté
        serializer.save(author=self.request.user)
```

→ Sans `perform_create`, DRF essaierait d'enregistrer l'article sans auteur, ce qui peut causer une erreur si le champ est obligatoire.

### ◆ Requête :

```
POST /articles/
{
  "title": "Mon premier article",
  "content": "Ceci est un article."
}
```

---

→ **Résultat** : L'article est créé avec `author=self.request.user` automatiquement.

---

## ◆ Exemple 2 : Modifier un champ avant sauvegarde

On veut s'assurer que le titre commence par une majuscule avant d'enregistrer l'objet :

```
class ArticleCreateView(CreateAPIView):
    queryset = Article.objects.all()
    serializer_class = ArticleSerializer

    def perform_create(self, serializer):
        title = serializer.validated_data.get("title", "")
        serializer.save(title=title.capitalize())
```

✓ Même si l'utilisateur envoie "titre en minuscule", l'article sera enregistré avec "Titre en minuscule".

---

## ◆ Exemple 3 : Effectuer une action après la création

On veut envoyer un email après la création d'un utilisateur :

```
from django.core.mail import send_mail

class UserCreateView(CreateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer

    def perform_create(self, serializer):
        user = serializer.save()
        # Envoyer un email de bienvenue
        send_mail(
            "Bienvenue sur notre site",
            f"Bonjour {user.username}, merci de vous être inscrit !",
            "admin@exemple.com",
            [user.email],
            fail_silently=False,
        )
```

---



## ◆ Résumé

Méthode	Rôle
<code>perform_create</code>	Personnalise la création d'un objet avant <code>serializer.save()</code> .

---

Méthode	Rôle
Cas d'utilisation	<ul style="list-style-type: none"><li>- Assigner <code>request.user</code> à un champ</li><li>- Modifier des valeurs avant sauvegarde</li><li>- Effectuer des actions après création (logs, email, etc.)</li></ul>

 **Obligatoire ?**  Non, DRF appelle `serializer.save()` par défaut.

 **Utile ?**  Oui, pour toute logique métier supplémentaire avant la création.

 **`perform_create` donne plus de contrôle sur la création d'objets !**