



Présentation de Django

Framework Web Python Haut-niveau

1. Introduction

- Framework web **Python**, haut-niveau
- Open-source
- Créé en **2005**, très actif
- Cas d'usage : APIs, backoffice, sites dynamiques

2. ⚙️ Architecture de Django

- Basé sur le pattern **MTV** :
 - **Model** : structure des données
 - **Template** : affichage HTML
 - **View** : logique métier
- Cycle HTTP :
 1. Requête → URL Dispatcher
 2. View → Modèle
 3. Template → Réponse

3. Composants principaux

a. ORM: Object-Relational Mapping

- Models déclarés via des classes Python
- Gèrent les tables SQL automatiquement

Exemple :

```
class Book(models.Model):  
    title = models.CharField(max_length=200)  
    author = models.CharField(max_length=100)  
    published_date = models.DateField()  
    isbn = models.CharField(max_length=13, unique=True)  
    cover = models.ImageField(blank=True)  
  
# Ensuite une requête simple  
Book.objects.filter(author="Django")
```

b. 🧠 Views

- Fonction ou classe
- Reçoit la requête, renvoie une réponse
- Exemple :

```
def hello(request):  
    return HttpResponse("Hello World")  
  
# Ou bien une classe  
  
class HelloWorldView(View):  
    def get(self, request):  
        return HttpResponse("Hello, World!")
```

c. Templates

- Moteur intégré Django: Jinja
- Syntaxe simple : `{% if %}`, `{% for %}`
- Insertion de variables avec `{{ var }}`

Exemple

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello</title>
</head>
<body>
  <h1>Liste des livres</h1>
  <ul>
    {% for book in books %}
      {% if book.pages > 100 %}
        <li><strong>{{ book.title }}</strong> ({{ book.pages }} pages)</li>
      {% else %}
        <li>{{ book.title }} (court)</li>
      {% endif %}
    {% empty %}
      <li>Aucun livre trouvé.</li>
    {% endfor %}
  </ul>
</body>
</html>
```


d. URLs & Routing

- Routage dans `urls.py`
- Paramètres dynamiques : `<int:id>`
- Utilisation de `include()` pour modulariser

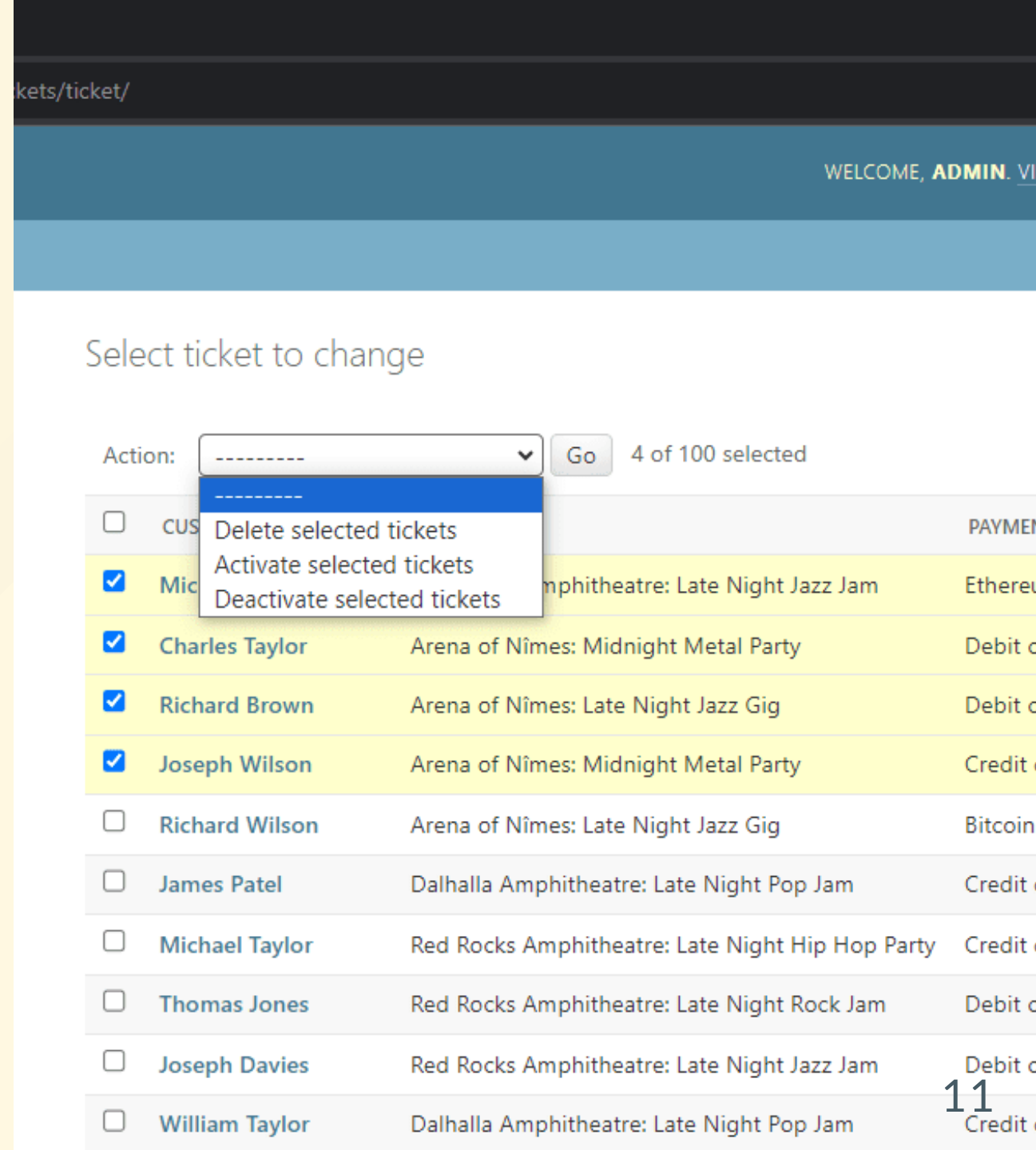
```
path("articles/<int:id>/", views.detail)
```

e. Formulaire & Validation

- `Form` = champs manuels
- `ModelForm` = automatique depuis un modèle
- Gestion des erreurs intégrée

f. Interface d'administration




- CRUD automatique depuis les modèles
- Interface prête à l'emploi
- Customisable avec `ModelAdmin`



4. 🚀 Fonctionnalités avancées

- **Middleware**s : traitement avant/après views
- **Signals** : déclencheurs sur événements
- Authentification, permissions
- **Caching & Logging**
- Support complet d'i18n (traductions)

5. Conclusion

-  Structure claire, rapide à mettre en œuvre
-  Excellente documentation (disponible en Français)
-  Communauté large
- Idéal pour les projets structurés (APIs, backoffice, SaaS)

 **Questions ?**

Merci pour votre attention !