Django : Configuration de la base de données

Enseignant:
M. Austin Waffo

LSI3- Dec 2023 - wkaustins@gmail.com

Configuration de la base de données

mysite/settings.py

Ouvrez mysite/settings.py

- définissez TIME_ZONE selon votre fuseau horaire

Configuration de la base de données

mysite/settings.py

Ouvrez mysite/settings.py

- définissez TIME_ZONE selon votre fuseau horaire
- Notez également le réglage INSTALLED_APPS qui contient par défaut :
  - django.contrib.admin
  - django.contrib.auth
  - django.contrib.contenttypes
  - django.contrib.sessions
  - django.contrib.messages
  - django.contrib.staticfiles

Configuration de la base de données

mysite/settings.py

Ouvrez mysite/settings.py

- définissez TIME_ZONE selon votre fuseau horaire
- Notez également le réglage INSTALLED_APPS qui contient par défaut :
  - django.contrib.admin
  - django.contrib.auth
  - django.contrib.contenttypes
  - django.contrib.sessions
  - django.contrib.messages
  - django.contrib.staticfiles

Configuration de la base de données

mysite/settings.py

Ouvrez mysite/settings.py

- définissez TIME_ZONE selon votre fuseau horaire
- Notez également le réglage INSTALLED_APPS qui contient par défaut :
  - django.contrib.admin
  - django.contrib.auth
  - django.contrib.contenttypes
  - django.contrib.sessions
  - django.contrib.messages
  - django.contrib.staticfiles

Configuration de la base de données

Exécuter la commande **python manage.py migrate** :

Configuration de la base de données

Exécuter la commande **python manage.py migrate** : La commande migrate examine le réglage INSTALLED_APPS et crée les tables de base de données nécessaires en fonction des réglages de base de données dans votre fichier mysite/settings.py et des migrations de base de données contenues dans l'application (nous les aborderons plus tard)

Création des modèles

Dans notre application de sondage, nous allons créer deux modèles :

- Question et Choice (choix)

Création des modèles

Création des modèles

Dans notre application de sondage, nous allons créer deux modèles :

- Question et Choice (choix)
- Une Question possède une question et une date de mise en ligne.
- Un choix a deux champs : le texte représentant le choix et le décompte

Création des modèles

Dans notre application de sondage, nous allons créer deux modèles :

- Question et Choice (choix)
- Une Question possède une question et une date de mise en ligne.
- Un choix a deux champs : le texte représentant le choix et le décompte des votes. Chaque choix est associé à une Question.

Création des modèles

Dans notre application de sondage, nous allons créer deux modèles :

- Question et Choice (choix)
- Une Question possède une question et une date de mise en ligne.
- Un choix a deux champs : le texte représentant le choix et le décompte des votes. Chaque choix est associé à une Question.

Ces concepts sont représentés par des classes Python. Éditez le fichier polls/models.py de façon à ce qu'il ressemble à ceci :

Création des modèles

polls/models.py

```python
from django.db import models


class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField("date published")


class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Figure – Création modèle

Activation des modèles

Indiquer au projet que l'application **polls** est installée

mysite/settings.py

```
INSTALLED_APPS = [
    "polls.apps.PollsConfig",
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
]
```

Figure – Création modèle

Activation des modèles

Indiquer au projet que l'application **polls** est installée

mysite/settings.py

```
INSTALLED_APPS = [
    "polls.apps.PollsConfig",
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
]
```

Figure – Création modèle

Activation des modèles

Activation des modèles

- Exécuter la commande **python manage.py makemigrations polls**

Activation des modèles

Activation des modèles

- Exécuter la commande **python manage.py makemigrations polls**
- Exécuter la commande **python manage.py sqlmigrate polls 0001**, pour accepter des noms de migrations et afficher le code SQL correspondant
- La commande sqlmigrate n'exécute pas réellement la migration dans votre base de données - elle se contente de l'afficher à l'écran de façon à vous permettre de voir le code SQL que Django pense nécessaire. C'est utile pour savoir ce que Django s'apprête à faire ou si vous avez des administrateurs de base de données qui exigent des scripts SQL pour faire les modifications.

Activation des modèles

Activation des modèles

- Exécuter la commande **python manage.py makemigrations polls**
- Exécuter la commande **python manage.py sqlmigrate polls 0001**, pour accepter des noms de migrations et afficher le code SQL correspondant
- La commande sqlmigrate n'exécute pas réellement la migration dans votre base de données - elle se contente de l'afficher à l'écran de façon à vous permettre de voir le code SQL que Django pense nécessaire. C'est utile pour savoir ce que Django s'apprête à faire ou si vous avez des administrateurs de base de données qui exigent des scripts SQL pour faire les modifications.
- Maintenant, exécutez à nouveau la commande migrate pour créer les tables des modèles dans votre base de

Activation des modèles

Activation des modèles

- Exécuter la commande **python manage.py makemigrations polls**
- Exécuter la commande **python manage.py sqlmigrate polls 0001**, pour accepter des noms de migrations et afficher le code SQL correspondant
- La commande sqlmigrate n'exécute pas réellement la migration dans votre base de données - elle se contente de l'afficher à l'écran de façon à vous permettre de voir le code SQL que Django pense nécessaire. C'est utile pour savoir ce que Django s'apprête à faire ou si vous avez des administrateurs de base de données qui exigent des scripts SQL pour faire les modifications.
- Maintenant, exécutez à nouveau la commande migrate pour créer les tables des modèles dans votre base de

Activation des modèles

Activation des modèles

- Exécuter la commande **python manage.py makemigrations polls**
- Exécuter la commande **python manage.py sqlmigrate polls 0001**, pour accepter des noms de migrations et afficher le code SQL correspondant
- La commande sqlmigrate n'exécute pas réellement la migration dans votre base de données - elle se contente de l'afficher à l'écran de façon à vous permettre de voir le code SQL que Django pense nécessaire. C'est utile pour savoir ce que Django s'apprête à faire ou si vous avez des administrateurs de base de données qui exigent des scripts SQL pour faire les modifications.
- Maintenant, exécutez à nouveau la commande migrate pour créer les tables des modèles dans votre base de

Jouer avec l'interface de programmation(API)

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Afficher les questions présentes dans le système : **Question.objects.all()**

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Afficher les questions présentes dans le système : **Question.objects.all()**
- from django.utils import timezone

Jouer avec l'interface de programmation(API)

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Afficher les questions présentes dans le système : **Question.objects.all()**
- from django.utils import timezone
- q = Question(question_text="What's new ?", pub_date=timezone.now())
- q.save()
- q.id

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Afficher les questions présentes dans le système : **Question.objects.all()**
- from django.utils import timezone
- q = Question(question_text="What's new ?", pub_date=timezone.now())
- q.save()
- q.id
- q.question_text

Jouer avec l'interface de programmation(API)

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Afficher les questions présentes dans le système : **Question.objects.all()**
- from django.utils import timezone
- q = Question(question_text="What's new ?", pub_date=timezone.now())
- q.save()
- q.id
- q.question_text
- q.question_text = "What's up ?"
- q.save()

Ajout de méthodes

Ajouter une méthode __str()__ dans le fichier **polls/models.py**

polls/models.py ¶

```
from django.db import models


class Question(models.Model):
    # ...
    def __str__(self):
        return self.question_text


class Choice(models.Model):
    # ...
    def __str__(self):
```

Ajout de méthodes

Ajouter une méthode __str()__ dans le fichier **polls/models.py**

polls/models.py ¶

```python
import datetime

from django.db import models
from django.utils import timezone


class Question(models.Model):
    # ...
    def was_published_recently(self):
        return self.pub_date >= timezone.now() - datetime.timedelta(days=1)
```

Figure – Ajouter de la méthode **was_published_recently()**

Jouer avec l'interface de programmation(API)

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Question.objects.all()

Jouer avec l'interface de programmation(API)

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Question.objects.all()
- from django.utils import timezone

Jouer avec l'interface de programmation(API)

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Question.objects.all()
- from django.utils import timezone
- current_year = timezone.now().year
- Question.objects.get(pub_date__year=current_year)

Jouer avec l'interface de programmation(API)

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Question.objects.all()
- from django.utils import timezone
- current_year = timezone.now().year
- Question.objects.get(pub_date__year=current_year)
- q.choice_set.all()

Jouer avec l'interface de programmation(API)

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Question.objects.all()
- from django.utils import timezone
- current_year = timezone.now().year
- Question.objects.get(pub_date__year=current_year)
- q.choice_set.all()
- Création de trois choix
- q.choice_set.create(choice_text="Not much", votes=0)
- q.choice_set.create(choice_text="The sky", votes=0)

Jouer avec l'interface de programmation(API)

Jouer avec l'interface de programmation(API)

Lancer le shell à travers la commande **python manage.py shell**. Exécuter respectivement les commandes suivantes :

- from polls.models import Choice, Question
- Question.objects.all()
- from django.utils import timezone
- current_year = timezone.now().year
- Question.objects.get(pub_date__year=current_year)
- q.choice_set.all()
- Création de trois choix
- q.choice_set.create(choice_text="Not much", votes=0)
- q.choice_set.create(choice_text="The sky", votes=0)
- q.choice_set.all()

Introduction au site d'administration de Django

Création d'un utilisateur administrateur

- **python manage.py createsuperuser**
- saisir ensuite : le **username**, **mail** et **mot de passe**

Introduction au site d'administration de Django

Création d'un utilisateur administrateur

- **python manage.py createsuperuser**
- saisir ensuite : le **username**, **mail** et **mot de passe**

Démarrage du serveur de développement

- **python manage.py runserver**
- saisir dans la barre d'adresse du navigateur **http ://127.0.0.1 :8000/admin/.**

Introduction au site d'administration de Django

Création d'un utilisateur administrateur

- **python manage.py createsuperuser**
- saisir ensuite : le **username**, **mail** et **mot de passe**

Démarrage du serveur de développement

- **python manage.py runserver**
- saisir dans la barre d'adresse du navigateur **http ://127.0.0.1 :8000/admin/.**

Rendre l'application de sondage modifiable via l'interface d'admin

Mais où est notre application de sondage ? Elle n'est pas affichée sur la page d'index de l'interface d'administration.

Rendre visible l'application de sondage

polls/admin.py ¶

```python
from django.contrib import admin

from .models import Question

admin.site.register(Question)
```

Figure – Rendre visible l'application

Dans la fenêtre d'administration, vous pouvez voir maintenant la liste des questions créées