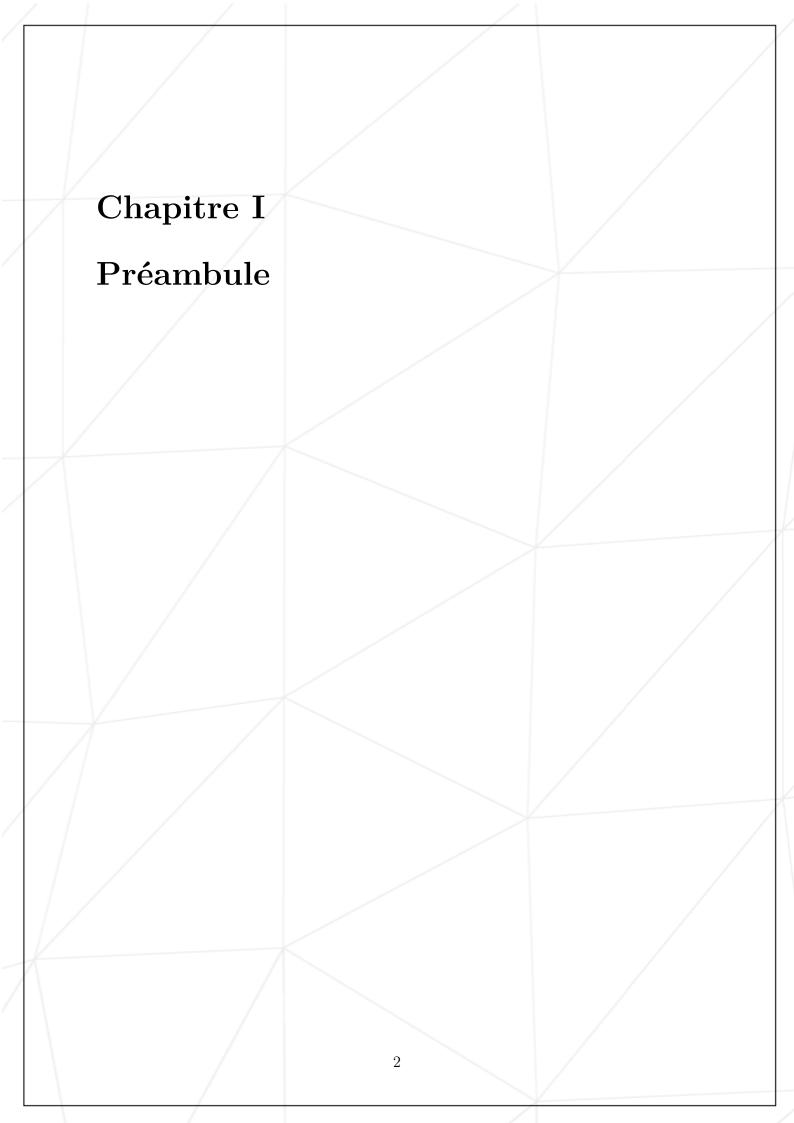


# D05 - Formation Python-Django Django - ORM

Résumé: Aujourd'hui nous allons découvrir l'ORM de Django.

# Table des matières

Ι	Préambule	2
II	Consignes	3
III	Règles spécifiques de la journée	4
IV	Exercice 00	6
V	Exercice 01	7
VI	Exercice 02	8
VII	Exercice 03	10
VIII	Exercice 04	12
IX	Exercice 05	14
$\mathbf{X}$	Exercice 06	16
XI	Exercice 07	18
XII	Exercice 08	20
XIII	Exercice 09	22
XIV	Exercice 10	24



### Chapitre II

# Consignes

Sauf contradiction explicite, les consignes suivantes seront valables pour tous les jours de cette Piscine.

- Seul ce sujet sert de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre <u>la procédure de rendu</u> pour tous vos exercices. L'url de votre dépot GIT pour cette journée est disponible sur votre intranet.
- Vos exercices seront évalués par vos camarades de Piscine.
- En plus de vos camarades, vous pouvez être évalués par un programme appelé la Moulinette. La Moulinette est très stricte dans sa notation car elle est totalement automatisée. Il est donc impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les mauvaises surprises.
- Les exercices shell doivent s'éxcuter avec /bin/sh.
- Vous <u>ne devez</u> laisser <u>aucun</u> autre fichier que ceux explicitement specifiés par les énoncés des exercices dans votre dépot de rendu.
- Vous avez une question? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Toutes les réponses à vos questions techniques se trouvent dans les man ou sur Internet.
- Pensez à discuter sur le forum Piscine de votre Intra et sur Slack!
- Lisez attentivement les exemples car ils peuvent vous permettre d'identifier un travail à réaliser qui n'est pas précisé dans le sujet à première vue.
- Réfléchissez. Par pitié, par Thor, par Odin!

## Chapitre III

# Règles spécifiques de la journée

- L'interprète à utiliser est python3.
- Chaque exercice est indépendant. Si parmi les features demandées, certaines ont déjà été réalisées dans les exercices précédents, dupliquez- les dans l'exercice courant.
- Vous devez travailler dans une base de données postgresql nommée formationdjango et créer un role nommé djangouser, dont le mot de passe est "secret", qui aura tous les droits dessus.
- Votre dossier de rendu doit être un projet Django. Le nom du projet doit être celui de la journée en cours.
- Nous allons utiliser le concept d'application de Django pour séparer les exercices : Chaque exercice de la journée doit se trouver dans une application Django distincte portant le nom de l'exercice correspondant et se trouvant à la racine du dossier de rendu, .
- Le projet Django doit être configuré correctement afin de remplir les conditions requises par les exercices. Aucune modification des configurations ne sera permise en soutenance.
- Vous ne devrez rendre aucune migration avec votre travail.
- Dans chaque exercice dont le cartouche mentionne ORM, vous devez exploiter l'ORM de Django. Aucune ligne de SQL ne doit être écrite.
- Dans chaque exercice dont le cartouche mentionne SQL, vous devez utiliser la librairie psycopg2 et effectuer toutes les requètes en SQL.

Voici un exemple de structure typique pour un rendu de l'étudiant krichard, concernant la journée d42 et comprenant deux exercices :

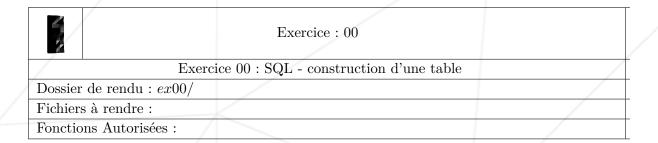
```
krichard
    .git
    .gitignore
    |-- __init__.py
    -- settings.py
    |-- urls.py
    |-- wsgi.py
    -- admin.py
    -- apps.py
     -- forms.py
       __init__.py
      - models.py
      - tests.py
     -- urls.py
     -- views.py
    ex01
    -- admin.py
      - apps.py
     - forms.py
       __init__.py
models.py
       tests.py
       urls.py
    -- views.py
    manage.py
```



Soyez malin : factorisez votre code et rendez le facile à réutiliser, vous gagnerez du temps.

### Chapitre IV

#### Exercice 00



Creez une application Django nommée ex00 ainsi qu'une vue à l'interieur de celle-ci qui doit être accessible via l'url : 127.0.0.1:8000/ex00/init.

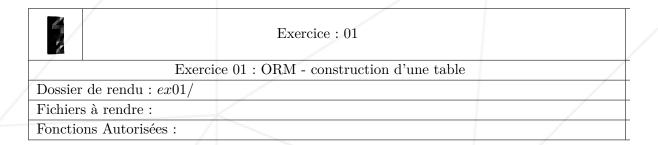
Cette vue doit créer une table SQL dans Postgresql à l'aide de la librairie psycopg2 et retourner une page contenant "OK" en cas de succes, ou dans le cas contraire un message d'erreur décrivant le problème rencontré.

La table SQL doit répondre a cette description. :

- Elle doit se nommer ex00 movies.
- Elle ne doit être créée que si elle n'existe pas déjà.
- Elle doit contenir uniquement les champs suivants :
  - o title : chaine de charactere variable, unique, d'une taille maximale de 64 octets, non nul.
  - o episode nb: entier, PRIMARY KEY.
  - o opening\_crawl: texte, peut être nul, pas de taille limite.
  - o director : chaîne de caractères variable, non nul, d'une taille maximale de 32 octets.
  - o producer : chaîne de caractères variable, non nul, d'une taille maximale de 128 octets.
  - o release\_date : date (sans l'heure), non nul.

# Chapitre V

#### Exercice 01



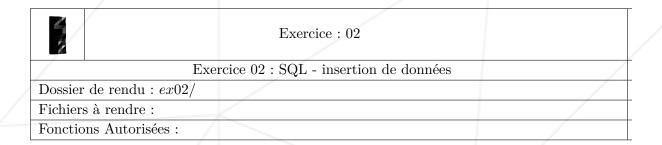
Creez une application nommée  $\tt ex01$ , et dans celle-ci un modèle  $\tt Django$  nommé  $\tt Movies$  ayant exactement ces champs :

- title : chaine de character, unique, d'une taille maximale de 64 octets, non nul.
- episode\_nb : entier, PRIMARY KEY.
- opening\_crawl : texte peut etre nul.
- director : chaîne de caractères d'une taille maximale de 32 octets, non nul.
- producer : chaîne de caractères d'une taille maximale de 128 octets, non nul.
- release\_date : date (sans l'heure), non nul.

Ce modèle doit également redéfinir la methode  $\_\mathtt{str}\_$  afin que celle-ci renvoie l'attribut  $\mathtt{title}$ 

# Chapitre VI

#### Exercice 02



Vous devez créer une application Django nommée ex02, et dans celle-ci, les vues accessibles via les urls suivantes :

• 127.0.0.1:8000/ex02/init : doit créer une table qui doit avoir en tout point les mêmes caracteristiques que celle demandées pour l'ex00 à la différence près qu'elle se nommera ex02\_movies.

Elle doit retourner une page affichant "OK" en cas de succès, ou dans le cas contraire un message d'erreur décrivant le problème rencontré.

- 127.0.0.1:8000/ex02/populate : doit insérer dans la table crée par la vue précédente, les données suivantes :
  - o episode\_nb : 1 title : The Phantom Menace director : George Lucas producer : Rick McCallum release\_date : 1999-05-19
  - o episode\_nb : 2 title : Attack of the Clones director : George Lucas producer : Rick McCallum release\_date : 2002-05-16
  - $\circ\,$ episode\_nb : 3 title : Revenge of the Sith director : George Lucas producer : Rick McCallum release\_date : 2005-05-19
  - o episode\_nb : 4 title : A New Hope director : George Lucas producer : Gary Kurtz, Rick McCallum release\_date : 1977-05-25
  - episode\_nb : 5 title : The Empire Strikes Back director : Irvin Kershner producer : Gary Kutz, Rick McCallum release\_date : 1980-05-17

- o episode\_nb : 6 title : Return of the Jedi director : Richard Marquand producer : Howard G. Kazanjian, George Lucas, Rick McCallum release\_date : 1983-05-25
- episode\_nb : 7 title : The Force Awakens director : J. J. Abrams producer : Kathleen Kennedy, J. J. Abrams, Bryan Burk release\_date : 2015-12-11

Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le nom du film, suivi du problème rencontré.

• 127.0.0.1:8000/ex02/display : doit afficher l'intégralité des données contenues dans la table ex02\_movies dans un tableau HTML y compris les éventuels champs nuls.

Si aucune donnée n'est disponible, ou en cas d'erreur, la page doit simplement afficher "No data available".

### Chapitre VII

#### Exercice 03

Exercice: 03	
Exercice 03 : ORM - insertion de données	
Dossier de rendu : $ex03/$	
Fichiers à rendre :	/
Fonctions Autorisées :	

Creez une nouvelle app Django nommé ex03 et créez dans celle-ci un modèle Django en tout point similaire à celui de l'ex01.

Cette app doit contenir les vues accessibles via les urls suivantes :

- 127.0.0.1:8000/ex03/populate : doit insérer, le modèle de cette application, les données suivantes :
  - o episode\_nb : 1 title : The Phantom Menace director : George Lucas producer : Rick McCallum release\_date : 1999-05-19
  - o episode\_nb : 2 title : Attack of the Clones director : George Lucas producer : Rick McCallum release date : 2002-05-16
  - $\circ$ episode\_nb : 3 title : Revenge of the Sith director : George Lucas producer : Rick McCallum release\_date : 2005-05-19
  - o episode\_nb : 4 title : A New Hope director : George Lucas producer : Gary Kurtz, Rick McCallum release\_date : 1977-05-25
  - episode\_nb : 5 title : The Empire Strikes Back director : Irvin Kershner producer : Gary Kutz, Rick McCallum release\_date : 1980-05-17
  - o episode\_nb : 6 title : Return of the Jedi director : Richard Marquand producer : Howard G. Kazanjian, George Lucas, Rick McCallum release\_date : 1983-05-25
  - $\circ$  episode\_nb : 7 title : The Force Awakens director : J. J. Abrams producer :

Kathleen Kennedy, J. J. Abrams, Bryan Burk - release\_date : 2015-12-11

Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontré.

• 127.0.0.1:8000/ex03/display : doit retourner une page HTML affichant l'intégralité des données contenues dans la table Movies, formatée dans un tableau HTML, y compris les éventuels champs nuls.

Si aucune donnée n'est disponible, la page doit simplement afficher "No data available".



En soutenance, la migration sera effectuée avant les tests.

## Chapitre VIII

#### Exercice 04

Exercice: 04	
Exercice 04 : SQL - suppression de donnée	
Dossier de rendu : $ex04/$	/
Fichiers à rendre :	/
Fonctions Autorisées :	

Creez une app nommée ex04. Elle doit contenir les vues accessibles via les urls suivantes :

• 127.0.0.1:8000/ex04/init : doit créer une table qui doit avoir en tout point les mêmes caractéristiques que celle demandées pour l'app l'ex00 à la différence près qu'elle se nommera ex04\_movies.

Elle doit retourner une page affichant "OK" en cas de succès, ou dans le cas contraire un message d'erreur décrivant le problème rencontré.

• 127.0.0.1:8000/ex04/populate : doit insérer, dans la table crée par la vue précédente, les données décrites dans l'exercice ex02.

Cette vue doit impérativement réinsérer toute donnée qui aurait été supprimée. Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontré.

• 127.0.0.1:8000/ex04/display : doit afficher l'intégralité des données contenue dans la table ex04\_movies dans un tableau HTML, y compris les éventuels champs nuls.

Si aucune donnée n'est disponible, ou en cas d'erreur la page doit simplement afficher "No data available".

• 127.0.0.1:8000/ex04/remove : doit afficher un formulaire HTML contenant une liste déroulante de titres de films et un bouton submit nommé remove.

Les titres de films sont ceux contenus dans la table  $ex04\_movies$ .

Lorsque le formulaire est validé, le film selectionné est supprimé de la base de données et le formulaire est réaffiché avec la liste de films restants mise à jour.

Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher "No data available".

### Chapitre IX

#### Exercice 05

	Exercice: 05	
Exercie	ce 05 : ORM - suppression de données	
Dossier de rendu : $ex05/$		
Fichiers à rendre :		
Fonctions Autorisées :		

Creez une nouvelle app Django nommé ex05 et créez dans celle-ci un modèle en tout point similaire à celui de l'ex01.

Cette app doit contenir les vues accessibles via les urls suivantes :

• 127.0.0.1:8000/ex05/populate : doit insérer, dans le modèle crée pour cette application, les données décrites dans l'exercice ex03.

Cette vue doit impérativement réinsérer toute donnée qui aurait été supprimée. Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontré.

• 127.0.0.1:8000/ex05/display : doit afficher l'intégralité des données contenues dans la table Movies de cette app dans un tableau HTML; y compris les éventuels champs nuls.

Si aucune donnée n'est disponible, ou en cas d'erreur, la page doit simplement afficher "No data available".

• 127.0.0.1:8000/ex05/remove : doit afficher un formulaire HTML contenant une liste déroulante de titres de films et un bouton submit nommé remove.

Les titres de films sont ceux contenus dans le modèle Movies de cette application.

Lorsque le formulaire est validé, le film selectionné est supprimé de la base de données et le formulaire est réaffiché avec la liste de films restants mise à jour.

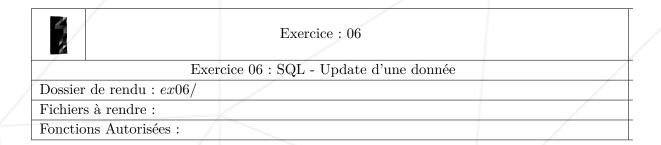
Si aucune donnée n'est disponible, la page doit simplement afficher "No data available".



En soutenance, la migration sera effectuée avant les tests.

### Chapitre X

#### Exercice 06



Creez une nouvelle app Django nommé ex06. Elle doit contenir les vues accessibles via les urls suivantes :

- 127.0.0.1:8000/ex06/init : doit créer une table qui sera exactement la même qu'à l'exercice 00 à la différence près qu'elle se nommera ex06\_movies et qu'elle disposera des champs supplementaires suivants :
  - o created de type datetime (date et heure), qui, à la création, doit être automatiquement assigné à la date et heure actuelle.
  - o updated de type datetime (date et heure), qui, à la création, doit être automatiquement assigné à la date et heure actuelle et automatiquement updaté à chaque mise à jour à l'aide du trigger suivant :

```
CREATE OR REPLACE FUNCTION update_changetimestamp_column()
RETURNS TRIGGER AS $$
BEGIN

NEW.updated = now();
NEW.created = OLD.created;
RETURN NEW;
END;
$$ language 'plpgsql';
CREATE TRIGGER update_films_changetimestamp BEFORE UPDATE
ON exO6_movies FOR EACH ROW EXECUTE PROCEDURE
update_changetimestamp_column();
```

• 127.0.0.1:8000/ex06/populate : doit peupler la table créée par la vue précédente avec les données décrites dans l'exercice ex02.

Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontré.

• 127.0.0.1:8000/ex06/display : doit afficher l'intégralité des données contenues dans la table ex06\_movies dans un tableau HTML.

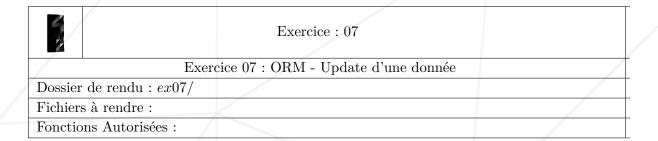
Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher "No data available".

• 127.0.0.1:8000/ex06/update : doit gérer l'envoi et la reception d'un formulaire. Ce dernier doit permettre de choisir un film dans une liste déroulante contenant les titres des films de la table ex06\_movies, et d'écrire du texte dans un deuxème champ. En validant le formulaire, la vue doit remplacer, dans la table ex06\_movies, le champ opening\_crawl du film sélectionné par le texte entré dans le formulaire.

Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher "No data available".

# Chapitre XI

#### Exercice 07



Creez une nouvelle app Django nommé ex07 et créez dans celle-ci un modèle en tout point similaire à celui de l'ex01, à la différence près que vous devez y rajouter les deux champs supplémentaires suivants :

- created de type datetime (date et heure), qui, à la création, doit être automatiquement assigné à la date et heure actuelle.
- updated de type datetime (date et heure), qui, à la création, doit être automatiquement assigné à la date et heure actuelle et automatiquement updaté à chaque mise à jour.

Cette app doit contenir les vues accessibles via les urls suivantes :

• 127.0.0.1:8000/ex07/populate : peuple le modèle crée précédemment avec les mêmes données qu'à l'ex02

Elle doit retourner une page affichant pour chaque insertion "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontrés.

• 127.0.0.1:8000/ex07/display : affiche l'intégralité des données contenues dans la table Movies dans un tableau HTML.

Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher "No data available".

• 127.0.0.1:8000/ex07/update : doit gérer l'envoie et la reception d'un formulaire. Ce dernier doit permettre de choisir un film dans une liste déroulante contenant les titres des films contenu dans le modèle Movies de cette application, et d'écrire du texte dans un deuxème champs.

En validant le formulaire, la vue doit modifier dans le modèle Movies de cette application, le champs opening\_crawl du film sélectionné avec le texte entré dans le formulaire.

Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher "No data available".



En soutenance, la migration sera effectuée avant les tests.

# Chapitre XII

#### Exercice 08

	Exercice: 08	
/	Exercice 08 : SQL - Foreign Key	
Dossier de rendu : $ex08/$		
Fichiers à rendre :		/
Fonctions Autorisées :		

Creez une nouvelle app Django nommé ex08. Cette app doit contenir les vues accessibles via les urls suivantes :

- 127.0.0.1:8000/ex08/init : Doit créer deux tables.

  La première doit se nommer ex08\_planets et avoir les champs suivants :
  - o id: serial, primary key
  - o name : chaîne de caratères variable, unique, non null, d'une taille maximale de 64.
  - o climate : chaîne de caratères variable.
  - o diameter: entier.
  - orbital\_period : entier.
  - population : gros entier.
  - rotation\_period : entier.
  - o surface water : réel.
  - o terrain : chaîne de caratères variable, d'une taille maximale de 128. La deuxième doit se nommer ex08\_people et avoir les champs suivants :
  - o id: serial, primary key.
  - o name : chaîne de caratères variable, d'une taille maximale de 64, unique, non null.

- o birth\_year : chaîne de caratères variable, d'une taille maximale de 32.
- o gender : chaîne de caratères variable, d'une taille maximale de 32.
- o eye\_color : chaîne de caratères variable, d'une taille maximale de 32.
- o hair\_color : chaîne de caratères variable, d'une taille maximale de 32.
- o height : entier.
- o mass : réel.
- o homeworld : chaîne de caratères variable, d'une taille maximale de 64, foreign key, référençant la colonne name de la table 08 planets
- 127.0.0.1:8000/ex08/populate : Doit peupler les deux tables en copiant le contenu des fichiers fournis people.csv et planets.csv dans les tables correspondantes, respectivement : ex08\_people et ex08\_planets.

  Cette vue doit retourner une page affichant pour chaque fichier "OK" en cas de succès, ou dans le cas contraire : un message décrivant le problème rencontré.
- 127.0.0.1:8000/ex08/display: affiche tous les noms de personnages, leur planète d'origine ainsi que le climat, dont ledit climat est tout ou partie venteux ('windy'), trié par ordre alphabetique des noms de personnage.

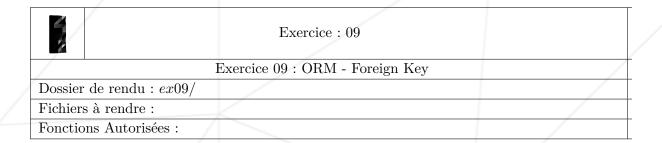
Si aucune donnée n'est disponible ou en cas d'erreur, la page doit simplement afficher "No data available".



Renseignez vous sur la methode copy\_from de psycopg2

# Chapitre XIII

#### Exercice 09



Creez une nouvelle app Django nommé ex09 et créez dans celle-ci deux modèles. Le premier modèle que vous devez créer doit se nommer Planets et doit contenir les champs suivant :

- name : chaîne de caractères, unique, non null, taille maximum de 64.
- climate : chaîne de caractères.
- diameter : entier.
- orbital\_period : entier.
- population : gros entier.
- rotation\_period : entier.
- surface water : réel.
- terrain : chaîne de caractères.
- created de type datetime (date et heure), qui, à la création, doit être automatiquement assigné à la date et heure actuelle.
- updated de type datetime (date et heure), qui, à la création, doit être automatiquement assigné à la date et heure actuelle et automatiquement updaté à chaque mise à jour.

Ce modèle doit également redéfinir la methode \_\_str\_\_() afin que celle-ci renvoie l'attribut name.

Le deuxième modèles que vous devez créer doit se nommer People et doit contenir les champs suivant :

- name : chaîne de caractères, max 64, unique, non nul.
- birth\_year : chaîne de caractères, max 32.
- gender : chaîne de caractères, max 32.
- eye\_color : chaîne de caractères, max 32.
- hair\_color : chaîne de caractères, max 32.
- height : entier.
- mass : réel.
- homeworld : chaîne de caractères, max 64, foreign key référençant la colonne name de la table Planets de cette application.
- created de type datetime (date et heure), qui, à la création, doit être automatiquement assigné à la date et heure actuelle.
- updated de type datetime (date et heure), qui, à la création, doit être automatiquement assigné à la date et heure actuelle et automatiquement updaté à chaque mise à jour.

Ce modèle doit également redéfinir la methode \_\_str\_\_() afin que celle-ci renvoie l'attribut name.

Créez également dans cette app, une vue qui doit être accessible à l'adresse 127.0.0.1:8000/ex09/display.

Cette vue doit afficher dans un tableau HTML tous les noms de personnages, leur planète d'origine ainsi que le climat, dont ledit climat est tout ou partie venteux ('windy'), trié par ordre alphabetique des noms de personnages.

Si aucune donnée n'est disponible, la vue doit afficher le texte "No data available, please use the following command line before use:" suivi d'une ligne de commande.

Cette ligne de commande doit être celle à exécuter depuis la racine de votre rendu afin d'insérer dans les modèles crées précédement, toutes les données présentes dans le fichier ex09\_initial\_data.json (fourni dans les ressources de la journée).

Vous devrez donc fournir ce fichier avec votre rendu.



En soutenance, la migration sera effectuée avant les tests.

# Chapitre XIV

#### Exercice 10

Exercice: 10	
Exercice 10 : ORM - Many to Many	
Dossier de rendu : $ex10/$	/
Fichiers à rendre :	/
Fonctions Autorisées :	

Creez une nouvelle app Django nommée ex10 et créez dans celle-ci trois modèles :

- Planets et People : Ces deux modèles doivent être identiques en tout point à ceux de l'exercice ex09.
- Movies: Ce modèle doit être identique en tout point à celui de ex01 à la différence près que vous devez y rajouter le champ characters.
   Ce champ est de type "many to many" avec le modèle People et il permet de répertorier tous les personnages présents dans un film et se trouvant dans la table People.

Les fixtures nécessaires au peuplement des modèles sont présents dans le fichier ex10 initial data.json fournis parmis les ressources de la journée.

Vous devez également créer dans cette app la vue accessible via l'url 127.0.0.1:8000/ex10. Celle-ci doit afficher un formulaire disposant de ces champs (chacuns obligatoire) :

- Movies minimum release date : date
- Movies maximum release date : date
- Planet diameter greater than : nombre
- Character gender : liste déroulante contenant les différentes valeurs disponibles dans le champs gender du modèle People. Il ne doit pas s'y trouver plusieurs fois la même valeur.

Une fois validée, la vue doit chercher, renvoyer et afficher le ou les résultats.

Un résultat est un personnages dont le genre correspond au champ 'character gender', avec un film dans lequel il est présent et dont la date de sortie se situe entre Movies minimum release date et Movies maximum release date, avec sa planète d'origine dont le diamètre est supérieur ou égale à Planet diameter greater than.

S'il n'existe aucun résultat correspondant à la requête, "Nothing corresponding to your research" doit s'afficher.

Chaque résultat doit s'afficher sur une ligne avec ces éléments (pas nécessairement dans cet ordre) :

- Le nom du personnage
- Son genre
- Le titre du film
- Le nom de sa planète d'origine (homeworld)
- Le diamètre de sa planete d'origine

Par exemple : les résultats pour des personnages de genre féminin, dont les films sont sortis entre "1900-01-01" et "2000-01-01" et dont la planète d'origine a un diamètre supérieur à 11000 sont :

- A New Hope Leia Organa female Alderaan 12500
- The Phantom Menace Padmé Amidala female Naboo 12120
- Return of the Jedi Leia Organa female Alderaan 12500
- Return of the Jedi Mon Mothma female Chandrila 13500
- The Empire Strikes Back -Leia Organa female Alderaan 12500



Plusieurs personnages peuvent se trouver dans un même film, et un même personnage peut apparaître dans plusieurs films. C'est ce qu'on appelle une relation many to many (plusieurs à plusieurs). Dans ce cas, une table intermédiaire doit être crée entre ces deux tables. Chaque rang de cette table intermédiaire est une association (unique) de deux références : la première référençant un rang de la table des films, la deuxième référençant un rang de la table des personnages (ou dans l'ordre inverse). Une fois vos modèles construits et vos migrations effectuées, vous pourrez voir cette table via la console de postgre.